Aide à la sélection de cibles pour des environnements de réalité virtuelle

Jonathan Wonner

N° d'ordre : 2262

THÈSE

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie
École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

présentée pour obtenir le grade de

Docteur de l'Université de Strasbourg

Mention Informatique

par

Jonathan Wonner

Aide à la sélection de cibles pour des environnements de réalité virtuelle

---

Soutenue publiquement le **16 décembre 2013** devant la commission d'examen composée de :

Mme Dominique Bechmann ..... *Directeure de thèse et Examinatrice*
Professeure à l'Université de Strasbourg

M. Jérôme Grosjean ..... *Encadrant et Examinateur*
Maître de conférence à l'Université de Strasbourg

M. Antonio Capobianco ..... *Encadrant et Examinateur*
Maître de conférence à l'Université de Strasbourg

Mme Indira Thouvenin ..... *Présidente du jury et Examinatrice*
Professeure à l'Université de Technologie de Compiègne

M. Bruno Arnaldi ..... *Rapporteur*
Professeur à l'INSA de Rennes

M. Paul Richard ..... *Rapporteur*
Maitre de conférence à l'Université d'Angers

ICube **UMR 7357 - CNRS - Université de Strasbourg**

La création est un acte de pure volonté.

John Hammond, *Jurassic Park*

Remerciements

Ça y est, c'est la fin de cette thèse, il est temps pour moi d'arrêter de hanter les couloirs à 6h du matin et de remercier les personnes qui m'ont entourées durant ces trois ans. Tout d'abord, je tiens à remercier Dominique Bechmann pour avoir accepté de diriger cette thèse après mes six mois de stage de M2. Jérôme Grosjean et Antonio Capobianco, que je remercie également pour m'avoir encadré, complètent ce trio de choc. J'ai ainsi pu profiter de leur expérience, de leurs conseils avisés et de leur enthousiasme pendant toute la durée de mes recherches, ainsi que pendant la rédaction de ce document. Merci encore à vous.

Je souhaite remercier les membres du jury, Indira Thouvenin, Bruno Arnaldi, et Paul Richard, qui ont accepté d'évaluer ce travail, pour leurs remarques constructives.

Je remercie également l'ensemble de l'équipe Informatique Géométrique et Graphique, pour la bonne humeur, l'ambiance chaleureuse et les nombreuses parties de tarot, à l'intérieur des locaux ou même en dehors. Toutes ces personnes m'ont permis de passer une thèse agréable pendant ces trois ans (et quelques mois). Je ne vais pas tous les citer, mais je remercie particulièrement Olivier (pour ses nombreux conseils et aides techniques, et pour nos longues discussions du matin), mon coubureau Rémi, et mes deux autres compagnons de thèse Kenneth et Lionel du bureau de la fête 2.0, toujours partants pour une free food ou une sortie détente (préparez-vous pour les Horror Nights 2014).

Je voudrais remercier mes amis en dehors du labo qui m'ont soutenu à chaque instant, en me réconfortant dans les moments de doute, et en me faisant passer des bons moments de détente. Je pense notamment à Jeannot (Belzébuth), Nico, Lulu, Arnaud, Amandine, Alexis, Adeline (vous sept, préparez-vous pour la raclette annuelle !), Gali (toujours prête pour aller boire un coup), Benoît (dieu, tout simplement), Aurore (toujours là malgré le décalage horaire, miss Caribou), Loïs (force de l'amitié), Philippe (notre maître à tous) ...

J'aimerais remercier ma famille pour m'avoir toujours encouragé, même si elle ne savait pas toujours ce que je faisais (déjà que moi-même...). Je dédie d'ailleurs cette thèse à ma grand-mère Mamounette, décédée récemment. Je remercie également ma belle-famille pour m'avoir souvent et si bien accueilli pendant mes vac... euh... mes changements d'en-

vironnements. Un énorme merci à vous tous.

Je ne pouvais pas finir ces remerciements sans évoquer Webcam, qui m'a appris que dormir est une activité futile après 5h, et surtout formellement interdite lorsque le bol de croquettes est vide. Grâce à elle, mon temps de travail a considérablement augmenté.

Enfin, mon plus grand remerciement va à ma fiancée Mylène. Son soutien sans faille m'a permis de surmonter les moments de doute et de peur. Sa présence à mes côtés a suffit à rendre agréables ces années de travail. Sans elle, je n'aurais pas pu aller au bout de cette thèse. Un gigantesque merci.

Table des matières

I Introduction et contexte **5**

1 Réalité virtuelle **7**
1.1 Introduction . . . 7
1.2 Applications de réalité virtuelle . . . 11
1.2.1 Simulation . . . 11
1.2.2 Applications médicales . . . 13
1.2.3 Fouille de données . . . 14
1.2.4 Jeux vidéo . . . 15

2 Les tâches fondamentales **18**
2.1 Contrôle d'application . . . 19
2.1.1 Dispositifs d'entrées . . . 19
2.1.2 Menus graphiques . . . 20
2.1.3 Reconnaissance de geste . . . 22
2.2 Manipulation . . . 23
2.2.1 Interaction libre . . . 24
2.2.2 Interaction contrainte . . . 24
2.3 Navigation . . . 25
2.3.1 Motivations . . . 27
2.3.2 Dispositifs matériels . . . 27
2.3.3 Techniques virtuelles . . . 29
2.4 Sélection . . . 30
2.5 Contexte . . . 31
2.6 Conclusion . . . 32

II Sélection : théorie et état de l'art **34**

3 Contexte théorique **36**
3.1 Problématiques de la sélection . . . 37
3.1.1 Loi de Fitts . . . 37

3.1.2 Stéréoscopie . . . 38
3.1.3 Densité et occlusions . . . 39
3.1.4 Tremblements naturels . . . 39
3.1.5 Bilan . . . 39
3.2 Phases de la sélection . . . 40
3.2.1 Phase de décision . . . 40
3.2.2 Phase balistique . . . 41
3.2.3 Phase de contrôle . . . 41
3.3 Conclusion . . . 41

4 État de l'art **44**
4.1 Localisation . . . 45
4.1.1 Techniques exocentriques . . . 45
4.1.2 Techniques égocentriques . . . 48
4.2 Prédiction . . . 52
4.3 Avatar de la main . . . 55
4.4 Rayons . . . 60
4.5 Aides proposées . . . 66
4.5.1 Localisation de cible . . . 66
4.5.2 Avatar de la main . . . 67
4.5.3 Prédiction de fin de mouvement . . . 67
4.6 Conclusion . . . 68

III Aides à la sélection **70**

5 Ring Concept et Bubble Bee **74**
5.1 Principe du Ring Concept . . . 76
5.2 Mise en œuvre du Bubble Bee . . . 78
5.3 Évaluation du Ring Concept . . . 80
5.3.1 Matériel et participants . . . 81
5.3.2 Procédure . . . 81
5.3.3 Conception . . . 83
5.3.4 Difficulté de la tâche . . . 84
5.4 Résultats . . . 85
5.4.1 Temps de décision . . . 85
5.4.2 Niveau d'erreur . . . 86
5.5 Discussion . . . 87
5.6 Conclusion . . . 88

6 Starfish **92**
6.1 Principe . . . 93
6.2 Introduction aux surfaces implicites . . . 96
6.3 Mise en œuvre . . . 99

6.3.1 Contrôle de Starfish . . . 99
6.3.2 Construction de la surface . . . 100
6.3.3 Contrainte du noyau . . . 110
6.3.4 Sélection de cible . . . 113
6.4 Évaluation de Starfish . . . 114
6.4.1 Matériel et participants . . . 114
6.4.2 Procédure . . . 115
6.4.3 Conception . . . 115
6.5 Résultats . . . 116
6.5.1 Temps de sélection . . . 116
6.5.2 Taux d'erreurs . . . 118
6.5.3 Questionnaire subjectif . . . 119
6.6 Discussion . . . 119
6.7 Conclusion . . . 121

7 SPEED : Speed Profile sEparation for Endpoint Divination 124
7.1 Profil de vitesse . . . 126
7.2 Algorithme SPEED . . . 127
7.2.1 Détection du pic . . . 128
7.2.2 Distance de prédiction . . . 129
7.2.3 Direction de prédiction . . . 131
7.2.4 Position de la prédiction . . . 132
7.2.5 Discussion sur les paramètres . . . 132
7.3 Évaluation de SPEED . . . 132
7.3.1 Matériel et participants . . . 133
7.3.2 Procédure . . . 133
7.4 Résultats et Discussion . . . 134
7.5 Conclusion . . . 136

8 Conclusion 138
8.1 Bilan . . . 138
8.2 Perspectives . . . 141
8.2.1 Ring Concept . . . 141
8.2.2 Starfish . . . 141
8.2.3 SPEED . . . 142

A Carré latin 161

B Méthode des moindres carrés 163

Table des figures

1 Utilisateur manipulant un objet virtuel sur un Workbench 2

1.1 Boucle des interfaces comportementales 9
1.2 La plateforme de réalité virtuelle IN VIRTUO 9
1.3 Périphériques d'interaction 10
1.4 Périphériques haptiques 10
1.5 Le système FIACRE de la SNCF 12
1.6 Simulateur d'arrêts de tirs de handball 12
1.7 LAP Mentor Virtual Reality Simulator 13
1.8 SpiderWorld 14
1.9 Visualisation 3D de données scientifiques 14
1.10 Immersion dans un jeu en réalité virtuelle 15
1.11 Visiocasques 16
1.12 Projections murales 16

2.1 Illustration d'un contrôle d'application par dispositif tactile mobile 20
2.2 Le Command& Control Cube 21
2.3 Spin Menu 22
2.4 Le système de menu TULIP 22
2.5 Reconnaissance de geste pour le contrôle d'application 23
2.6 Création d'une surface par un pinceau 3D 24
2.7 Positionnement d'un objet par une surface tactile 25
2.8 Réduction des degrés de liberté en décomposant la tâche 26
2.9 CrOS : déplacement géodésique sur la surface d'un objet 26
2.10 Présentation d'un site disparu 27
2.11 Dispositifs matériels pour la navigation 28
2.12 Virtuix Omni^{TM} 29
2.13 Techniques virtuelles pour la navigation 30
2.14 Maillages volumiques 31

3.1 Expérience de « va-et-vient » conduite par Paul Fitts 38

4.1 Aperçu d'un radar 2D 46
4.2 Utilisation du radar dans les jeux vidéo 46

4.3 World In Miniature . . . 47
4.4 Flower Garden . . . 48
4.5 Aperçu d'une flèche 3D . . . 49
4.6 Ailettes sur les flèches 3D . . . 49
4.7 Cluster de flèches . . . 50
4.8 Aroundplot . . . 50
4.9 City Lights . . . 51
4.10 Wedge . . . 51
4.11 Pic de vélocité dans un profil de vitesse . . . 53
4.12 KEP : Modélisation du profil de vitesse . . . 54
4.13 L'application « Use-the-force » avec un BCI. . . . 54
4.14 Utilisation d'un BCI pour la prédiction de trajectoire chez un primate. . . . 55
4.15 Principe de la technique Go-Go. . . . 56
4.16 Version 3D du Bubble Cursor . . . 57
4.17 Balloon Selection en action . . . 58
4.18 Dessin 3D avec une grille d'aimants haptiques . . . 58
4.19 Zones d'influence des aimants triples . . . 59
4.20 Champ de potentiel associé à un objet de CAO . . . 59
4.21 Le cône virtuel de FOLLOW-ME . . . 60
4.22 Mise en place de HardBorders . . . 61
4.23 La technique Head Crusher . . . 62
4.24 La technique Voodoo Dolls . . . 62
4.25 Comparaison entre un rayon, un cône et IntenSelect . . . 63
4.26 Utilisation du Depth Ray . . . 64
4.27 Procédure du SQUAD . . . 65
4.28 Illustration de la technique Expand . . . 65
4.29 Discordance de la visibilité . . . 66

5.1 Perception de la longueur d'un segment . . . 76
5.2 Orientation d'un cercle selon son demi-petit axe . . . 77
5.3 Redondance de l'information directionnelle . . . 78
5.4 Code couleur du Ring Concept . . . 78
5.5 Bubble Bee . . . 79
5.6 Scène de l'évaluation du Bubble Bee . . . 80
5.7 Illustration du niveau d'erreur . . . 82
5.8 Taille effective des cibles . . . 85
5.9 La Teapot Bee . . . 89
5.10 Le widget d'orientation . . . 90

6.1 Comportement du pointeur selon sa position . . . 94
6.2 Contraintes à base de cônes . . . 95
6.3 Exemple de surfaces isopotentielles . . . 96
6.4 Mélange de champs radiaux . . . 97
6.5 Champ généré par un segment . . . 98

6.6 Projection d'un point sur un segment . . . 99
6.7 Calcul de distance à un segment . . . 99
6.8 Contrôle du Starfish . . . 100
6.9 Déplacement du noyau sur la surface . . . 101
6.10 Filtre d'angle pour le squelette de Starfish . . . 102
6.11 Champ de potentiel pour un segment . . . 103
6.12 Surface de Starfish . . . 104
6.13 Surface générée par un squelette à 4 segments . . . 107
6.14 Visuel de Starfish sans Ring Concept . . . 108
6.15 Visuel de Starfish . . . 108
6.16 Visuel de Starfish en $3D$ . . . 109
6.17 Saut entre deux branches . . . 111
6.18 Comportement du pointeur captif . . . 111
6.19 Starfish : Scènes présentées pendant l'expérience . . . 114
6.20 Starfish : Diagramme en boîte pour le temps de sélection . . . 117
6.21 Starfish : Diagramme en barres pour le taux d'erreurs . . . 118
6.22 Starfish : Résultats subjectifs . . . 120

7.1 Décomposition d'un profil de vitesse . . . 127
7.2 Modélisation des deux phases du mouvmement . . . 127
7.3 SPEED : Détection du pic global . . . 129
7.4 SPEED : Mauvaise estimation du modèle quadratique . . . 130
7.5 SPEED : Calcul de la distance de prédiction . . . 131
7.6 SPEED : Cas de dépassement de cible . . . 131
7.7 SPEED : Construction du point de prédiction . . . 132
7.8 SPEED : Représentation de l'expérience . . . 133
7.9 SPEED : Comparaison des taux de réussite de SPEED et KEP . . . 135

Introduction

> This is where the fun begins.
>
> Anakin Skywalker,
> *Star Wars Episode III*
> *Revenge of the Sith*

La réalité virtuelle. Deux termes qui semblent s'opposer. La *réalité*, d'une part, s'accroche à notre monde physique, dans lequel nous vivons. Le *virtuel* caractérise une entité qui n'a pas d'existence tangible. Depuis l'avancée de la technologie informatique, ce deuxième terme est couramment employé dans le sens de *numérique*. L'expression *réalité virtuelle* réunit donc le monde physique et le monde numérique pour désigner un espace réaliste en trois dimensions, calculé en temps réel par un système informatique, dans lequel un utilisateur est immergé pour interagir avec lui. Il ne s'agit donc pas d'un monde réel, mais d'un monde numérique possédant certaines propriétés du monde réel. Cet environnement étant créé artificiellement, il peut être inspiré d'un environnement existant dans le monde physique, ou complètement imaginaire. Sur la figure 1, un utilisateur est immergé dans une scène le confrontant à un objet n'existant que dans le monde numérique. À l'aide de périphériques dédiés, l'utilisateur peut interagir avec cet objet et les autres entités de ce monde.

Dans la vie de tous les jours, nous sommes habitués à utiliser des applications numériques, comme un éditeur de texte ou un navigateur Web, et ce, sur de multiples supports, comme une station de travail classique, une tablette tactile, un smartphone ou une console de jeu. La plupart de ces applications requièrent une interaction uniquement bidimensionnelle. Lorsque l'application exploite un monde en trois dimensions, comme un simulateur de vol, l'interaction, elle, se fait toujours dans deux dimensions. Par exemple, la souris d'un ordinateur se déplace sur le plan de notre bureau, et le smartphone limite notre espace d'interaction à sa surface tactile. Les logiciels de modélisation 3D, comme Blender, s'adaptent à cette contrainte en proposant à l'utilisateur plusieurs points de vue de la même scène, l'une d'elle permettant de visualiser l'objet modélisé en trois dimensions, les autres étant des projections de cet objet sur des plans. Dans le contexte de la réalité virtuelle, les applications sont enrichies d'une troisième dimension, la profondeur, au niveau de la visualisation grâce au principe de stéréoscopie, et au niveau de l'interaction à l'aide de périphériques dédiés. L'ajout de cette troisième dimension

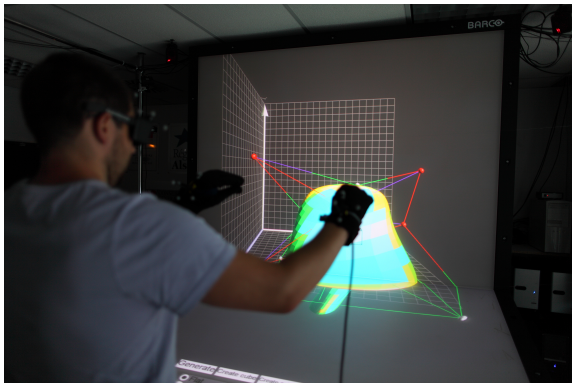


FIGURE 1 – Utilisateur manipulant un objet virtuel sur un Workbench, un dispositif de réalité virtuelle affichant un environnement immersif, à l'aide de gants de données, un périphérique d'interaction. Image extraite de [Wonner 11a].

semble être particulièrement adapté aux tâches de nature intrinsèquement 3D, comme le positionnement d'un objet dans l'espace. Là où il faudrait déplacer l'objet dans trois vues différentes dans une application 2D, une manipulation directe est possible en réalité virtuelle.

Toutefois, si les environnements virtuels ont développé une nouvelle méthode d'interaction, ils ont aussi engendré de nouvelles problématiques. La profondeur reste, malgré l'utilisation de la stéréoscopie, la dimension la plus difficile à appréhender, et il est parfois difficile d'interpréter correctement la scène, en particulier la position des objets les uns par rapport aux autres, et par rapport à l'utilisateur. Le simple fait de désigner un objet spécifique, tâche relativement simple dans une application 2D à l'aide d'une souris, est soumis à de nombreux obstacles. La scène peut en effet être suffisamment dense pour que l'objet d'intérêt soit dissimulé par les autres entités. Positionner précisément notre main dans l'espace est également plus difficile et plus fatiguant, du fait des tremblements naturels engendrés par l'absence de support physique, que lorsqu'elle est posée sur le support d'une souris. Si la manipulation d'objets reste la tâche principale de la plupart des applications de réalité virtuelle, la sélection est une étape préliminaire et indispensable à la manipulation. Le développement de techniques aidant l'utilisateur dans cette tâche pour améliorer ses performances semble nécessaire.

L'objectif de notre travail est de proposer des outils d'interaction permettant de palier certains obstacles à la sélection en environnement 3D immersif. Parmi ces obstacles, nous retrouvons les problématiques liées à la localisation d'une cible d'intérêt dans la scène, et à la précision de l'interaction 3D. Dans la partie I de ce document, nous présentons les enjeux de la réalité virtuelle. Après une présentation des types d'applications existants, nous distinguons et décrivons les tâches fondamentales pouvant être réalisées

dans ces environnements : le contrôle d'application, la manipulation, la navigation et la sélection. Dans ce document, nous nous intéressons plus particulièrement à la tâche de sélection, qui est au cœur de notre travail. La partie II est consacrée à cette tâche fondamentale et préliminaire à la manipulation. Nous présentons dans un premier temps les aspects théoriques de la sélection. Nous détaillons en particulier les différentes étapes composant cette tâche, ainsi que les différentes problématiques qui lui sont liées. Dans un second temps, nous proposons un état de l'art des différentes techniques développées, qui ont permis d'améliorer les performances de l'utilisateur pour cette tâche. L'étude de ces techniques nous a conduit à nous interroger sur le développement de nouvelles approches pour aborder la sélection. Dans la partie III, nous présentons les trois nouvelles techniques que nous avons mis en œuvre dans le cadre de notre travail. Le **Ring Concept**, au chapitre 5, est une indication visuelle permettant de désigner la position d'un objet d'intérêt en pointant la direction dans laquelle il se trouve. Cette méthode a été conçue afin de pouvoir être intégrée visuellement à d'autres outils manipulés par l'utilisateur, évitant ainsi la multiplication des outils affichés. Nous avons ainsi implémenté le **Bubble Bee**, provenant de l'augmentation d'une technique de sélection, le 3D Bubble Cursor, par le Ring Concept. Dans le chapitre 6, **Starfish** est une technique de sélection exploitant des guides virtuels pour amener le pointeur de l'utilisateur précisément sur sa cible. Ces guides prennent la forme de branches définies implicitement, et dont la forme s'adapte dynamiquement afin de capturer les cibles proches du pointeur. Enfin, dans le chapitre 7, **SPEED** est un algorithme modélisant le profil de vitesse d'un mouvement de sélection en deux dimensions afin d'estimer la position de la cible de l'utilisateur avant qu'il ne l'atteigne. Dans ces trois chapitres, nous décrivons nos motivations et détaillons le principe de nos nouveaux outils. Nous concevons également un protocole expérimental mettant en avant l'aide fournie par ces techniques, en comparaison avec les techniques existantes. Nous finissons ce document par quelques perspectives que nous proposons dans le cadre de ce travail.

Première partie

Introduction et contexte

Chapitre 1

Réalité virtuelle

Welcome... to the real world.

Morpheus,
The Matrix

Sommaire

1.1 Introduction 7
1.2 Applications de réalité virtuelle 11
1.2.1 Simulation 11
1.2.2 Applications médicales 13
1.2.3 Fouille de données 14
1.2.4 Jeux vidéo 15

Dans ce document, nous nous intéressons spécifiquement à la tâche de sélection dans les environnements de réalité virtuelle. Ces environnements sont différents à bien des égards des stations de travail classique que nous utilisons quotidiennement via des dispositifs comme un clavier et une souris. Nous présentons dans ce premier chapitre ce concept de réalité virtuelle. Nous expliquons dans un premier temps sa définition formelle telle qu'elle est indiquée dans le Traité de la Réalité Virtuelle [Fuchs 03], puis nous donnons un aperçu de la plateforme matérielle que nous avons exploitée dans le cadre de cette thèse. Dans un second temps, nous présentons un éventail des applications de la réalité virtuelle et des domaines de recherche liés à cette technologie.

1.1 Introduction

Le Traité de la Réalité Virtuelle [Fuchs 03, Fuchs 06] donne la définition suivante de la réalité virtuelle :

Définition 1 *La réalité virtuelle est un domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3D, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensori-moteurs.*

De cette définition, nous retenons que la réalité virtuelle permet à un utilisateur d'agir sur un environnement virtuel. Le système interprète les actions de l'utilisateur pour modifier l'environnement, évalue les informations sensorielles (images, efforts, sons, odeurs, ...) à transmettre dues à ces modifications, et enfin transmet ces informations aux interfaces. Nous pouvons donc dire qu'à l'instar de notre monde réel, l'utilisateur agit sur l'environnement virtuel, et inversement l'environnement virtuel agit sur l'utilisateur. Le terme de *réalité artificielle* semble donc refléter plus précisément ce concept [Krueger 91], même si l'expression *réalité virtuelle* reste largement la plus employée.

Interfaces comportementales

Une application basée sur les techniques de la réalité virtuelle nécessite l'utilisation d'interfaces comportementales. Nous en distinguons trois catégories. Les interfaces sensorielles transmettent des stimuli du système vers l'utilisateur, comme du son ou un retour haptique. À l'inverse, les interfaces motrices informent le système des mouvements de l'utilisateur et de ses actions sur le monde virtuel. Enfin, les interfaces sensori-motrices transmettent les informations dans les deux sens. Le choix du nombre et du type de ces interfaces dépend de l'application et de son objectif. Cependant, la grande majorité des environnements de réalité virtuelle comporte un système de visualisation, le plus souvent stéréoscopique afin d'immerger l'utilisateur à l'intérieur du monde virtuel. Le système HOMERE [Lécuyer 03] en est un contre-exemple et propose un système haptique et non visuel pour explorer l'environnement, à l'attention des utilisateurs malvoyants. La figure 1.1 représente les échanges d'informations entre l'utilisateur et le monde virtuel par l'intermédiaire des interfaces comportementales.

Interfaces sensorielles

Dans le cadre de cette thèse, nous avons profité de la plateforme de réalité virtuelle IN VIRTUO — *l'expérience virtuelle* du laboratoire ICube [ICube]. Cette plateforme s'appuie principalement sur un plan de travail virtuel immersif (Workbench) et d'un mur immersif (voir figure 1.2). Ces deux systèmes de visualisation disposent d'écrans à stéréoscopie active - deux écrans de 2m de diagonale pour le Workbench, et un de 3.7m de diagonale pour le mur immersif - et intègrent un système de capture de mouvements de l'utilisateur.

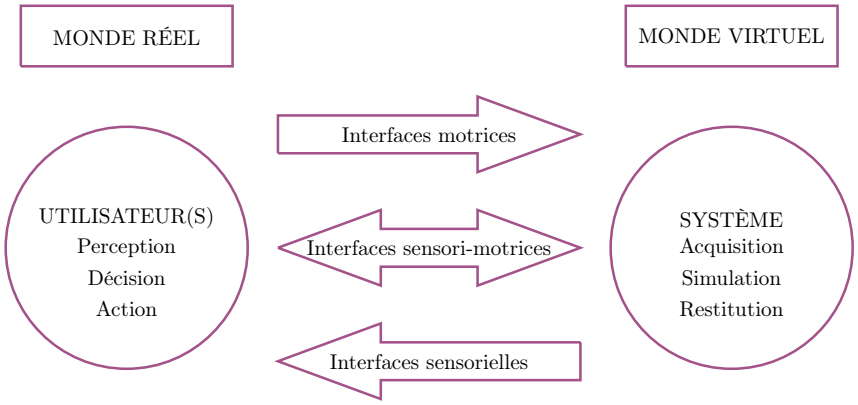


FIGURE 1.1 – La boucle d'échange entre l'utilisateur et le système par les interfaces comportementales. Schéma inspiré de [Fuchs 06].

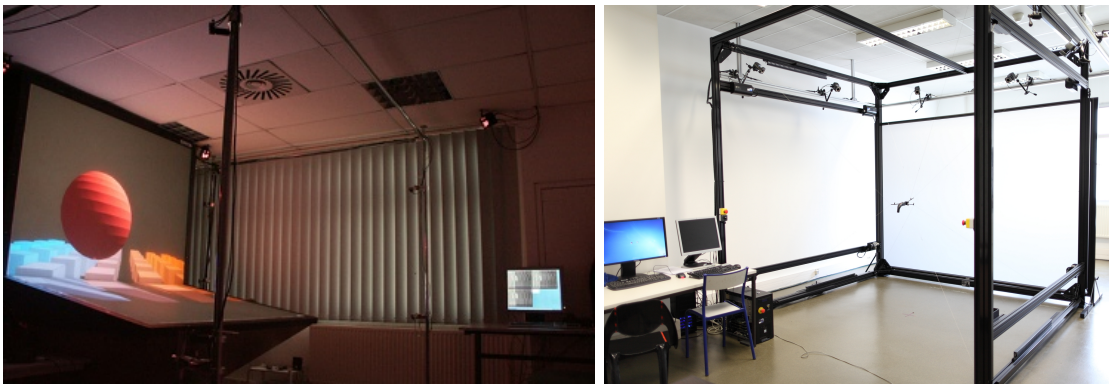


FIGURE 1.2 – Les systèmes de visualisation de la plateforme de réalité virtuelle IN VIRTUO. **À gauche :** le Workbench. **À droite :** le mur immersif.

Interfaces motrices

À ces systèmes de visualisation sont intégrés des périphériques permettant à l'utilisateur d'agir dans l'environnement virtuel, un *flystick*, une *Wiimote* et des gants de données. Un *flystick* et une *Wiimote* de Nintendo® sont deux périphériques dotés de boutons que l'utilisateur peut actionner. Les gants de données sont des gants capables de détecter et mesurer les flexions et les contacts des doigts. Ces trois périphériques, montrés en figure 1.3, sont munis de réflecteurs infrarouges détectables par les systèmes de capture de mouvements, afin de détecter les positions et les orientations des mains de l'utilisateur.

FIGURE 1.3 – Les périphériques d'interaction de la plateforme de réalité virtuelle IN VIRTUO. **De gauche à droite :** le flystick, la Wiimote et les gants de données.

Interfaces sensori-motrices

Le mur immersif dont nous disposons est combiné à un périphérique à retour d'efforts de grande taille - robot à câbles Inca 6D de la société Haption [Haption]. Ce périphérique permet à l'utilisateur de percevoir un retour haptique (force et couple), et au système de mesurer les efforts fournis par l'utilisateur, à l'aide de huit câbles tendus reliés à des blocs moteurs et à une poignée, visibles sur la figure 1.2. La poignée est manipulée par l'utilisateur, tandis que les câbles s'enroulent et se déroulent en fonction du mouvement de la poignée. Les câbles peuvent également être immobilisés par les moteurs pour offrir une résistance à l'utilisateur.

Enfin, des périphériques haptiques à destination d'un usage sur station de travail classique sont également disponibles. Il s'agit de périphériques Phantom de la société SensAble [Geomagic], à 3 et 6 degrés de libertés (voir figure 1.4). L'utilisateur manipule un stylet relié à un bras mécanique percevant les efforts de l'utilisateur, et pouvant appliquer des forces au stylet.

FIGURE 1.4 – Les périphériques haptiques Phantom (Omni et Desktop) de la plateforme de réalité virtuelle IN VIRTUO.

Nous n'avons cité dans cette section que les interfaces dont nous disposons au sein

de notre plateforme. De nombreux autres périphériques existants sont présentés dans l'ouvrage *3D User Interface - Theory and Practice* [Bowman 04]. Bien évidemment, la création de nouvelles interfaces est intimement liée aux applications développées dans ces environnements. Dans la section suivante, nous présentons un aperçu des applications de la réalité virtuelle et des domaines de recherche qui y sont attachés.

1.2 Applications de réalité virtuelle

Lister l'ensemble des applications existantes exploitant la réalité virtuelle serait une tâche fastidieuse et nécessairement non exhaustive, du fait de l'essor actuel de ce domaine. Un grand nombre de ces applications est néanmoins présenté dans le volume 4 *Applications de la réalité virtuelle* du Traité de la réalité virtuelle [Arnaldi 06]. Nous citons ici brièvement quelques grands domaines d'applications.

1.2.1 Simulation

La réalité virtuelle se prête naturellement à la simulation du monde réel. Recréer virtuellement notre monde a plusieurs intérêts. Il est par exemple possible de créer de nouveaux scénarios à partir d'une situation donnée, afin de prédire le comportement des utilisateurs si ces scénarios se déroulaient dans le monde réel. Par exemple, nous pouvons simuler un incident dans une centrale nucléaire, et observer la réaction des employés immergés dans la centrale virtuelle. Ce scénario est souvent impossible à recréer volontairement dans le monde réel, tandis que sa simulation dans l'environnement virtuel est rejouable à souhait. Une application évidente de ces simulations est la formation des utilisateurs à accomplir une certaine tâche. Dans l'exemple de la centrale, les employés peuvent ainsi être formés à réagir correctement en cas d'incident. Concrètement, le système FIACRE, illustrée en figure 1.5, forme des agents de la SNCF à intervenir sur les voies ferrés [Lourdeaux 01]. Ce simulateur permet de présenter rapidement et à peu de frais de nombreuses situations aux agents.

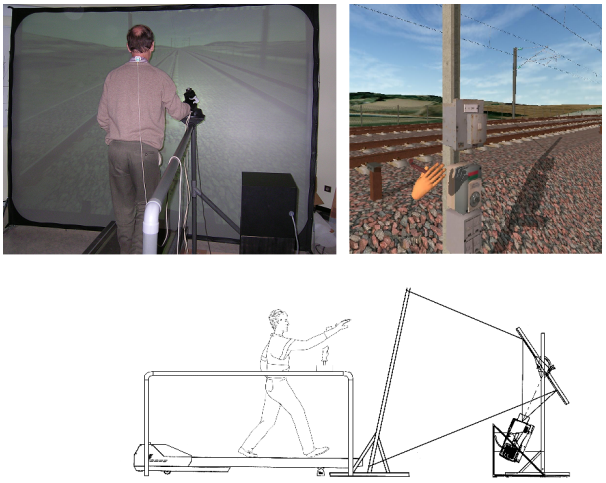


FIGURE 1.5 – Le simulateur FIACRE forme les agents de la SNCF à intervenir sur les voies ferrés. L'utilisation de ce système entraîne un gain de temps, et est plus rentable financièrement.

Le domaine sportif profite également de la réalité virtuelle, par exemple pour l'entraînement de lancers en basketball [Covaci 13], ou l'analyse des mouvements des sportifs et de leurs réactions face à différentes situations [Bideau 10]. La figure 1.6 montre un dispositif permettant à un gardien réel d'arrêter les tirs d'un joueur de handball virtuel. Ces simulateurs permettent d'anticiper et de se préparer à des situations spécifiques, et de corriger les erreurs des sportifs.

FIGURE 1.6 – Analyse des réactions d'un gardien de handball réel face à un lanceur virtuel. Image tirée de [Bideau 10].

1.2.2 Applications médicales

Le domaine médical exploite aussi les simulateurs afin de préparer par exemple des chirurgiens à exécuter le bon geste lors d'une opération délicate, ou à des fins d'entraînements et d'apprentissage pour des chirurgiens novices [Perrenot 12]. La figure 1.7 illustre l'un de ces simulateurs, le LAP Mentor Virtual Reality Simulator, pour une chirurgie laparoscopique [Bharathan 13] (sur la figure, une opération d'ablation des trompes). Dans ce type d'application, l'ajout d'une composante haptique est souhaitable pour refléter la résistance des différents organes [Gélinas-Phaneuf 13, Våpenstad 13]. Cependant, même si ces dispositifs permettent un entraînement illimité sur de nombreux cas différents, la réalité virtuelle semble n'être profitable que pour des chirurgiens novices, les plus expérimentés préférant l'utilisation de vrais corps [Sharma 12]. Des améliorations semblent en effet nécessaires concernant le réalisme et la réaction des organes lors d'opérations complexes.

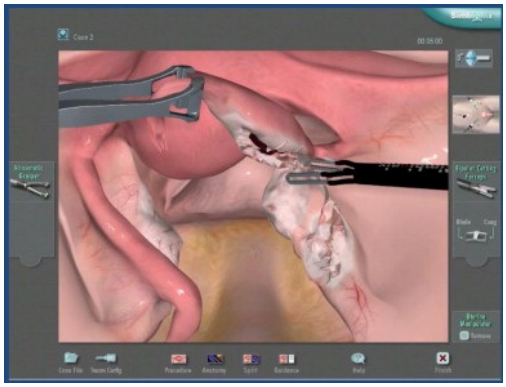


FIGURE 1.7 – Illustration du LAP Mentor Virtual Reality Simulator, un outil d'entraînement pour des opérations laparoscopiques. Image tirée de [Bharathan 13].

La psychothérapie bénéficie également de la réalité virtuelle dans le traitement des troubles anxieux, en particulier les phobies. En effet, le thérapeute peut exposer son patient à des situations anxiogènes de manière progressive et contrôlée [Gerardi 10]. Plusieurs études ont ainsi été réalisées et ont montré une baisse significative de l'anxiété après un tel traitement, par exemple pour l'acrophobie (peur du vide) [Coelho 09], l'arachnophobie (peur des araignées) [Carlin 97, Bouchard 06] (illustrée [1] en figure 1.8), ou encore l'aérodromophobie (peur des l'avion) [Rus-Calafell 13]. Ces thérapies sont souvent utiles dans la mesure où la qualité graphique ne semble pas être un facteur limitant, le scénario étant un facteur plus important que le réalisme pour placer le patient dans un état d'anxiété [Huem Kwon 13]. Cependant, la réalité virtuelle ne semble pas adaptée au traitement de toutes les phobies, comme l'agoraphobie [Meyerbroeker 13].

1. La recherche d'une image illustrant cette phobie a été particulièrement éprouvante. Je devrais peut-être suivre ce traitement.

FIGURE 1.8 – Illustration du SpiderWorld, exposant le patient à des araignées. Image tirée de [FirsthandTechnology].

1.2.3 Fouille de données

De nombreux domaines scientifiques profitent des avancées technologiques pour générer des bases de données toujours plus grandes. La fouille de données permet de déduire des relations dans ces bases. Ces relations sont principalement calculées par un ordinateur, bien que des analystes, en observant les données, puissent guider les calculs. La visualisation et l'interaction en environnement immersif, à l'aide de graphes 3D par exemple, permet d'observer de l'intérieur ces bases de données complexes [Nagel 08] et aide l'analyste à obtenir des relations. Sur la figure 1.9, un utilisateur observe l'hypocentre de tremblements de terre afin de déduire les zones à risques [Ogi 09].

FIGURE 1.9 – Visualisation de la relation entre des données de terrains et l'hypocentre de tremblements de terre.

1.2.4 Jeux vidéo

Depuis quelques années, le grand public s'intéresse à une interaction plus active et plus naturelle dans le jeu vidéo. Le jeu *Wii Sports*, par exemple, demande au joueur d'effectuer le même geste pour lancer une boule de bowling que dans le monde réel. Ce type d'interaction s'est largement développé avec le *Microsoft Kinect*, qui détecte les mouvements de l'utilisateur. L'utilisation d'une manette n'est ainsi plus nécessaire pour jouer. Il suffit au joueur d'effectuer les mouvements qu'il effectuerait dans le monde réel. Bien sûr, ce système n'est pas adapté à tous les types de jeu, et la plupart des jeux compatibles avec la *Kinect* sont des jeux de sport ou de danse. La réalité virtuelle permet ici d'immerger le joueur dans un monde 3D, comme illustrée en figure 1.10. Ce contrôle plus naturel demande un effort physique plus conséquent et entraîne une perte de performance. Néanmoins, les joueurs préfèrent la version immersive d'un jeu de type FPS[2] [Lugrin 13]. En effet, le réalisme graphique et la fidélité de l'interaction [McMahan 12] semblent être des facteurs plus importants que les performances de jeu dans le plaisir de jouer [Klimmt 09].

FIGURE 1.10 – Version immersive du jeu Unreal®Tournament 3. Image tirée de [Lugrin 13].

Le prix d'un dispositif de réalité virtuelle étant souvent un frein à son utilisation [Navarro 13], le domaine du jeu contribue à l'essor de la réalité virtuelle en motivant la création de nouveaux périphériques moins coûteux, en particulier des systèmes d'affichage comme des visiocasques (ou head mounted display, voir la figure 1.11) ou des projections murales (voir la figure 1.12).

2. *First person shooter* : jeu de tir en vue subjective

FIGURE 1.11 – Visiocasques. **Gauche :** Le modèle HMZ-T3W de *Sony*. Image tirée de [Sony]. **Droite :** L'Oculus Rift de *Oculus VR*. Image tirée de [Oculus VR].

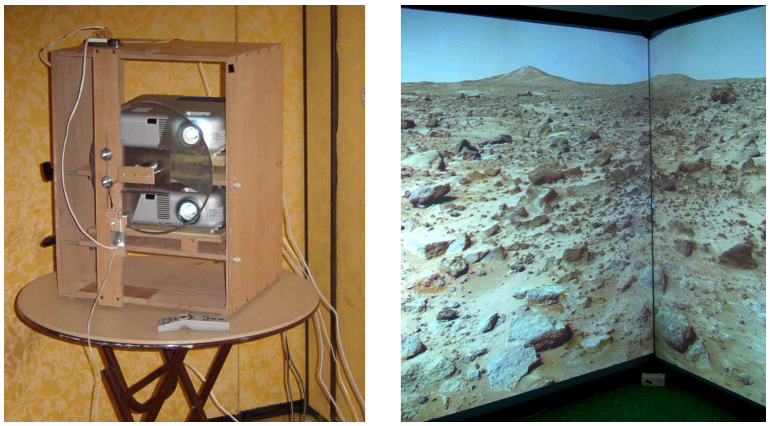


FIGURE 1.12 – Projections stéréoscopiques *low-cost* et multi-murales. Images tirées de [Miller 05].

Chapitre 2

Les tâches fondamentales

> Eternal glory, that is what awaits the student who wins the tri-wizard tournament. But to do this that student must survive three tasks. Three extremely dangerous tasks.
>
> Albus Percival Wulfric Brian Dumbledore,
> *Harry Potter and the Goblet of Fire*

Sommaire

2.1 Contrôle d'application 19
2.1.1 Dispositifs d'entrées 19
2.1.2 Menus graphiques 20
2.1.3 Reconnaissance de geste 22
2.2 Manipulation 23
2.2.1 Interaction libre 24
2.2.2 Interaction contrainte 24
2.3 Navigation 25
2.3.1 Motivations 27
2.3.2 Dispositifs matériels 27
2.3.3 Techniques virtuelles 29
2.4 Sélection 30
2.5 Contexte 31
2.6 Conclusion 32

Nous avons vu précédemment qu'il existe de nombreux types d'application prenant place dans des environnements de réalité virtuelle. La grande majorité de ces applications demande à l'utilisateur d'interagir avec le monde virtuel. Toutes les actions et tâches que

l'utilisateur peut effectuer peuvent être regroupées en quatre tâches fondamentales, définies par Bowman *et al.* [Bowman 99b, Bowman 04] : *contrôle d'application*, *manipulation*, *navigation* et *sélection*. Le contrôle d'application permet à l'utilisateur de communiquer avec le système en lui envoyant des commandes, afin d'accéder à ses fonctionnalités. La manipulation correspond à la modification et au repositionnement des objets de la scène. La navigation déplace l'utilisateur dans le monde virtuel. Enfin, la sélection est l'étape préliminaire à la tâche de manipulation, et permet de désigner un objet de la scène avant d'interagir avec lui.

Dans ce chapitre, nous présentons ces quatre tâches fondamentales de la réalité virtuelle, en décrivant les techniques utilisées pour réaliser ces tâches dans les applications existantes. L'objectif de ce chapitre est moins de proposer un état de l'art complet sur ces différentes techniques, que de décrire les quatre tâches fondamentales et de présenter succinctement les grandes familles des solutions existantes pour effectuer ces tâches. Notre travail portant spécifiquement sur la tâche de sélection, la partie II sera consacrée à cette tâche, et détaillera plus en avant ses tenants et aboutissants.

2.1 Contrôle d'application

Dans la grande majorité des applications 2D, nous retrouvons des éléments d'interface comme les boutons et les menus, qui permettent de contrôler l'application en envoyant des commandes au système, comme ouvrir et sauvegarder un fichier, modifier un paramètre, changer de mode, ... Bien que le « vrai » travail soit effectué par d'autres types d'interaction, comme la sélection et la manipulation d'objet à la souris, ou l'entrée de symboles (souvent au clavier), l'interface de contrôle est la clé de voûte qui permet à l'utilisateur de faire le lien entre ces tâches basiques et les fonctionnalités de l'application. À l'instar de leurs équivalents 2D que nous utilisons quotidiennement sur des stations de travail classiques, les applications 3D nécessitent elles-aussi une interface de contrôle.

2.1.1 Dispositifs d'entrées

La plupart des techniques de contrôle d'application nécessite un dispositif physique. Le flystick, une manette dont la position dans l'espace est généralement repérée à l'aide d'un système de tracking, est le plus couramment utilisé. Les boutons sur le flystick sont liés à une commande envoyée à l'application, pour changer de mode, valider une sélection, ouvrir un menu... De manière générale, les outils contrôlés par la main de l'utilisateur sont les plus adaptés pour cette tâche, bien qu'il soit possible d'employer un microphone pour un contrôle vocal. L'utilisation de gants de données autorise le contrôle d'application à l'aide d'une reconnaissance des gestes de la main. Certains gants contiennent des capteurs de pression aux extrémités des doigts, permettant également de considérer n'importe quelle surface comme tactile [Veit 11].

Si de nombreuses recherches s'emploient à développer de nouvelles techniques de contrôle 3D, l'essor des dispositifs tactiles mobiles (smartphones, tablettes) permet également d'utiliser les interfaces 2D sur ces périphériques. Une application pourrait par exemple afficher la scène 3D dans l'environnement virtuel, et un menu 2D « classique » sur le smartphone. Cette approche évite les problèmes d'occlusion entre l'interface et la scène, et permet d'exploiter les éléments connus comme un menu ou un bouton. La figure 2.1 représente la collaboration de deux utilisateurs dans un environnement 3D, l'un utilisant une technique de sélection basée sur un rayon, l'autre effectuant un contrôle à l'aide d'un dispositif tactile [Steinicke 08]. Cependant, une limitation de cette méthode est la nécessité de relâcher son attention de la scène 3D pour utiliser l'interface 2D, non intégrée à la scène.

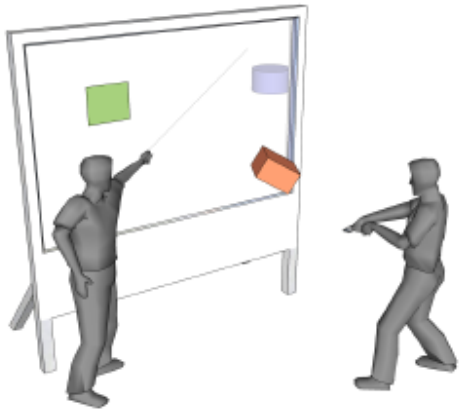


FIGURE 2.1 – Contrôle d'application par dispositif tactile mobile. L'utilisateur de gauche sélectionne un objet à l'aide d'un rayon virtuel, tandis que l'utilisateur de droite envoie une commande au système à l'aide d'un dispositif tactile. Image tirée de [Steinicke 08].

2.1.2 Menus graphiques

Les menus étant une interface très largement utilisée et efficace dans les interfaces 2D, il paraît naturel d'en développer une version 3D pour les environnements immersifs. Une solution simple consiste à afficher un menu 2D flottant dans l'espace 3D. Cette solution est intuitive car les utilisateurs, habitués aux interfaces 2D, la reconnaissent instantanément comme étant un élément d'interface. Cependant, il peut être malaisé de sélectionner un élément de ce menu à l'aide d'une technique de sélection 3D (voir la section 2.4 et le chapitre 4, consacrés à ces techniques de sélection).

Dans un environnement en trois dimensions, il semble naturel d'utiliser des éléments d'interface également en trois dimensions. Le C^3 (ou *Command & Control Cube*), illustré en figure 2.2, est un menu dont les éléments sont répartis dans une grille cubique de taille 3 [Grosjean 01]. Pour éviter les occlusions entre les éléments, seul un étage à la fois (soit 9 éléments) est affiché.

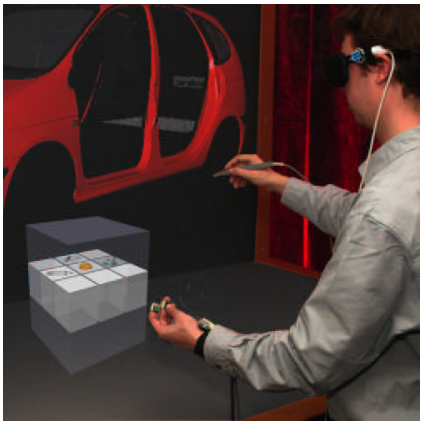


FIGURE 2.2 – Utilisation du C^3 (*Command & Control Cube*), un menu 3D dont les éléments sont répartis dans un cube de taille $3 \times 3 \times 3$. Image tirée de [Grosjean 03].

Si le menu est hiérarchique (avec des sous-menus), la question de la disposition des différents items est primordiale pour une interaction simple et rapide. Le *Spin Menu* est un menu dont les éléments sont placés selon un arc de cercle horizontal (voir la figure 2.3) [Gerber 04, Gerber 05]. Chaque élément est visible grâce à cette disposition en une dimension, et est accessible facilement par une rotation de la main. Après sélection d'un élément, un sous-menu peut être créé en affichant un autre arc de cercle verticalement, ou en dessous du précédent.

L'apparition d'un menu recouvre une partie de la scène virtuelle, et ce d'autant plus que le nombre d'éléments dans ce menu est important. Le système TULIP (*Three-Up, Labels In Palm*) a été développé pour résoudre cette contrainte [Bowman 01]. Lorsqu'il est activé, ce menu apparaît au niveau des mains de l'utilisateur, trois éléments étant placés sur l'index, le majeur et l'annulaire. Pour sélectionner un de ces éléments, l'utilisateur pince son pouce avec le doigt correspondant. Si le menu contient plus de trois éléments, l'utilisateur pince son pouce avec son auriculaire, pour faire apparaître les éléments suivants. Cette technique, illustrée en figure 2.4, limite la taille du menu et permet à l'utilisateur de contrôler sa position, afin de le gêner le moins possible dans sa tâche.

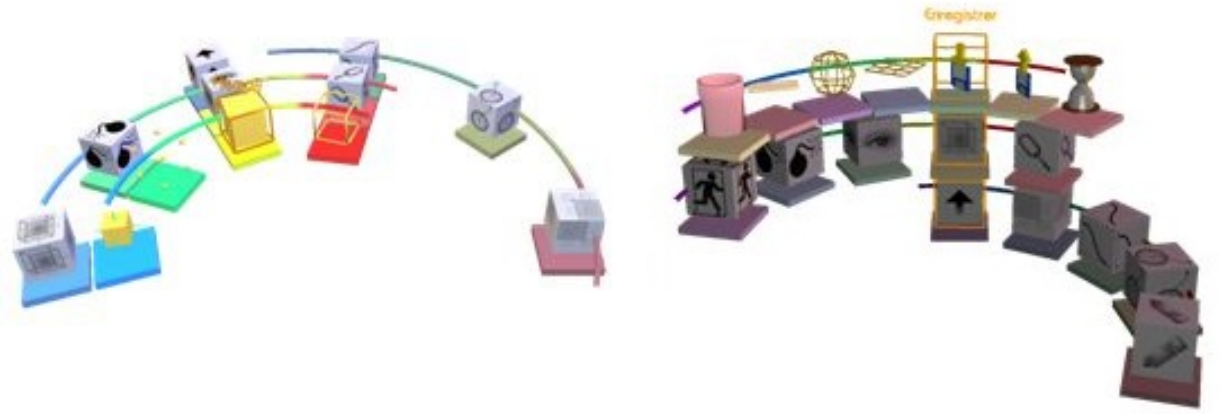


FIGURE 2.3 – Le Spin Menu, un menu circulaire hiérarchique. L'utilisateur peut manipuler le menu par des rotations de la main. Images tirées de [Gerber 05].

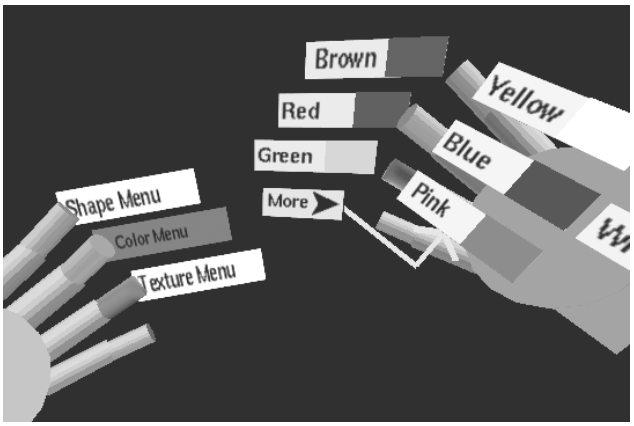


FIGURE 2.4 – Utilisation du TULIP, un menu dont les éléments sont répartis sur les doigts. Pour sélectionner un élément, l'utilisateur pince son pouce avec le doigt correspondant à cet élément. Image tirée de [Bowman 01].

2.1.3 Reconnaissance de geste

Dans les environnements de réalité virtuelle, la position et l'orientation de la tête de l'utilisateur sont enregistrées par un système de tracking, afin d'adapter l'image affichée en fonction de son point de vue. Ce système est également employé pour la position des mains. Il est en effet courant de lier une technique graphique, comme un menu ou une technique de sélection de type rayon, à la main de l'utilisateur. De plus, les gestes effectués peuvent être directement interprétés par le système comme une commande de contrôle. La technique SKETCH reconnaît de nombreux modèles de geste dans le cadre d'une

application de dessin [Zeleznik 96]. En un seul geste, l'utilisateur peut créer des formes, les modeler, les sélectionner, changer leur couleur, ... Cette solution souffre néanmoins de plusieurs contraintes. Les modèles de gestes doivent être suffisamment intuitifs et simples à retenir, comme le montre la figure 2.5. Il faut également s'assurer que les mains soient toujours visibles depuis le système de tracking, et que le système soit suffisamment robuste pour interpréter correctement les gestes exécutés.

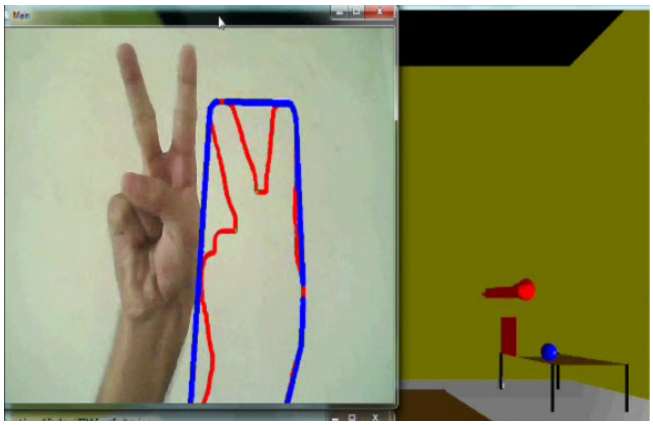


FIGURE 2.5 – Reconnaissance de geste pour le contrôle d'application. Ce modèle demande au système de déplacer l'objet sélectionné vers le haut. Image tirée de [Rautaray 12].

2.2 Manipulation

Bien que de nombreuses applications ne permettent qu'une exploration non interactive du monde 3D, la plupart des environnements de réalité virtuelle autorise une interaction avec les objets de la scène. Dans une application de CAO, un utilisateur peut créer des objets, les déplacer et les orienter, afin de les assembler ensemble. Dans une application de sculpture, l'utilisateur déforme un objet virtuel avec ses mains. Ces interactions sont des exemples de manipulation. Cette tâche est étroitement liée à la notion de degrés de liberté (noté DDL). Les caractéristiques principales d'un dispositif d'interaction sont le nombre de DDL que cet outil permet de contrôler, et le nombre de DDL pouvant être contrôlés simultanément. La souris classique permet un contrôle de 2 DDL, tandis qu'un flystick en autorise 6 (3 en translation et 3 en rotation). Les dispositifs obtenant les meilleurs performances dans les tâches de manipulation sont ceux intégrant le plus de DDL, car ils permettent une interaction similaire à une manipulation dans le monde réel [Zhai 97]. La plupart des techniques de manipulation suppose donc que l'utilisateur dispose d'un tel matériel.

2.2.1 Interaction libre

L'approche qui paraît la plus naturelle pour manipuler dans un espace 3D est de laisser l'utilisateur libre de ses mouvements. Ainsi, la technique du pinceau 3D permet de créer directement une surface en passant sa main où l'on souhaite qu'elle apparaisse [Schkolne 01], comme l'illustre la figure 2.6. La création d'objets plus complexes passe par la modification de ces surfaces par une technique de manipulation directe : l'utilisateur peut déformer une surface en la touchant. Une autre approche de la manipulation libre consiste à laisser l'utilisateur modifier la position des sommets d'un maillage, en les attrapant directement avec la main. Dans l'application THRED, l'utilisateur débute avec une surface carrée, qu'il est possible de subdiviser et de modifier en déplaçant les sommets de la surface [Shaw 94]. Ce type de technique, bien que basé sur un contrôle naturel des objets de la scène, ne permet pas un contrôle aussi précis que dans le monde réel.

FIGURE 2.6 – Création d'une surface par un pinceau 3D. Image tirée de [Schkolne 01].

2.2.2 Interaction contrainte

Le problème de précision induit par l'interaction libre incite l'introduction de contraintes dans les techniques de manipulation. Ces contraintes peuvent être de différentes natures. La technique VLEGO, utilisée dans une application d'assemblage de blocs géométriques [Kiyokawa 96], contraint ces blocs sur une grille à l'aide d'une technique de *snap-to-grid*. Les positions des sommets des blocs sont limitées à un ensemble discret de points disposés dans une grille régulière. Ceci permet de conserver une manipulation libre mais précise. Dans ce cas, les contraintes sont explicites (la grille discrète est visible) et uniquement visuelles. À l'inverse, la technique JDCAD exploite des contraintes implicites, dépendant de la position du pointeur de l'utilisateur [Liang 94]. Ainsi, si le pointeur se trouve près du centre d'une extrémité d'un cylindre, par exemple, l'utilisateur ne pourra que modifier la hauteur de ce cylindre. Ce type de contrainte limite les erreurs dues à la précision du geste de l'utilisateur.

Les techniques décrites précédemment contraignent les interactions entre l'utilisateur et les objets de la scène virtuelle. Une autre approche consiste à contraindre directement les mouvements de l'utilisateur, par exemple à l'aide d'une surface tactile comme illustrée en figure 2.7. L'utilisateur manipule dans les différents points de vue pour positionner l'objet dans l'espace 3D [Martinet 10]. Le mouvement est contraint sur le plan 2D de la surface tactile. Les périphériques haptiques sont un autre type de dispositif contraignant l'utilisateur. Le retour d'effort fourni par ce matériel donne à l'utilisateur la sensation de toucher physiquement les objets de la scène, et permet également à l'utilisateur d'appliquer une force sur un objet, améliorant le réalisme de l'interaction.

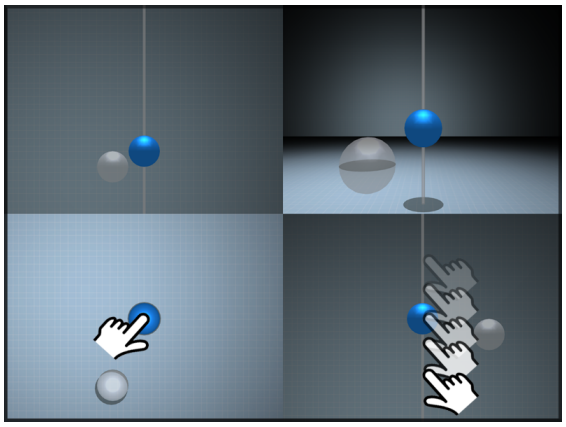


FIGURE 2.7 – Positionnement d'un objet par une surface tactile. L'utilisateur précise la position dans les différents points de vue. Image tirée de [Martinet 10].

Plus généralement, la réduction des degrés de liberté permet à l'utilisateur d'avoir un meilleur contrôle lors d'une tâche de manipulation. Ainsi, le positionnement d'un objet dans l'espace est effectué plus précisément si la tâche est décomposée en sous-tâches selon une ou deux dimensions (voir la figure 2.8), que lorsque l'interaction est non contrainte [Veit 11]. Ici, la réduction des degrés de libertés se traduit par une décomposition de la tâche. Une autre approche est de contraindre les mouvements de l'utilisateur tout en gardant une interaction 3D. La technique *CrOS* propose de lier le mouvement 2D de la main de l'utilisateur sur une surface tactile, à un déplacement géodésique sur le surface d'un objet [Veit 12] (voir la figure 2.9). Lorsque le pointeur sort du champ de vision de l'utilisateur, une réorientation automatique de l'objet est effectuée afin de faciliter la tâche.

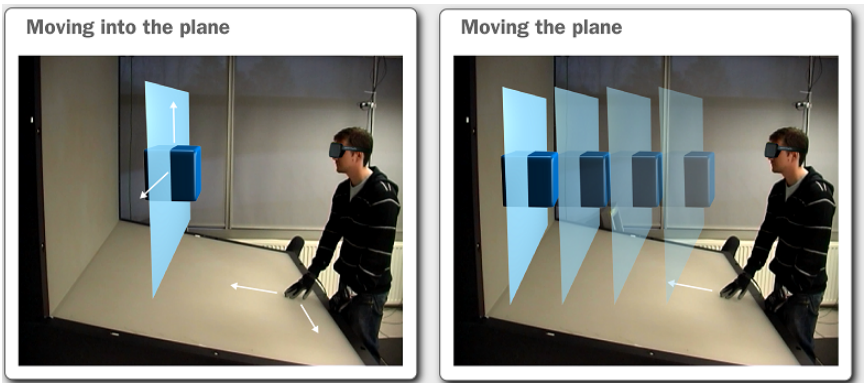


FIGURE 2.8 – Positionnement d'un objet dans l'espace en décomposant les degrés de liberté. L'utilisateur positionne d'abord l'objet selon un plan vertical, puis règle la profondeur. Image tirée de [Veit 11].

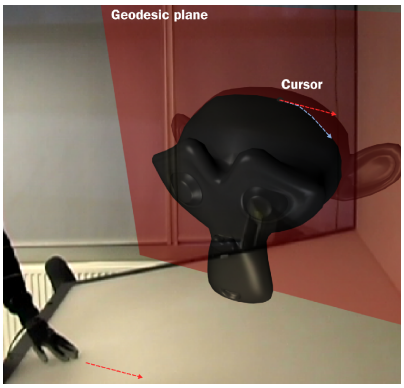


FIGURE 2.9 – Utilisation de la technique CrOS. L'utilisateur déplace le pointeur sur la surface de l'objet. Le pointeur est toujours visible grâce à une réorientation automatique. Image tirée de [Veit 12].

2.3 Navigation

Dans le monde réel, se déplacer ne nécessite pas une réflexion consciente permanente. Une fois que nous avons émis le souhait de traverser la pièce pour atteindre une porte, notre cerveau transmet aux muscles les mouvements corrects à effectuer, sans aide consciente de notre part, de la même façon que nous n'avons pas besoin de réfléchir pour respirer. Néanmoins, pour effectuer des déplacements plus complexes ou plus longs, nous utilisons des véhicules, comme une voiture ou un avion. Ces véhicules représentent une interface entre nos actions (tourner un volant ou appuyer sur une pédale) et le fait de naviguer. Dans un environnement immersif virtuel, notre capacité à se déplacer physique-

ment est limitée car la plupart des dispositifs de réalité virtuelle demande de garder une position debout et fixe, et nous employons donc également des interfaces pour voyager dans la scène.

2.3.1 Motivations

Avant de décrire certaines interfaces utilisées dans ce contexte, nous pouvons nous demander les raisons qui poussent un utilisateur à naviguer dans une scène. Certaines applications proposent uniquement une tâche d'exploration, où le seul objectif est de visiter la scène. Un architecte peut par exemple proposer une visite virtuelle du bâtiment qu'il a conçu, avant sa création. Si cette application permet d'explorer une scène qui n'existe pas encore dans le monde réel, de nombreuses applications sont consacrées à des scènes du passé, qui n'existent plus. La réalité virtuelle permet de diffuser de manière immersive un héritage culturel, comme l'illustre la figure 2.10. Dans le domaine du jeu vidéo, l'exploration du monde prend souvent une part importante du jeu, et est une étape indispensable afin de remplir les autres tâches.

FIGURE 2.10 – Présentation d'un site disparu à l'aide d'un Immersadesk®. Image tirée de [Gaitatzes 01].

À l'exploration libre d'un environnement virtuel s'oppose la tâche de recherche. L'utilisateur navigue dans le monde pour trouver un objet ou atteindre un lieu spécifique. Nous pouvons distinguer deux sous-tâches, selon que l'utilisateur connaisse la position de sa cible dans le monde, ou non. Dans le premier cas, l'utilisateur peut directement s'y rendre à l'aide de l'interface de navigation. Dans le second cas, la tâche se rapproche de la tâche d'exploration, car l'utilisateur doit révéler progressivement l'environnement encore inconnu, l'exploration s'arrêtant lorsque l'objet recherché est atteint.

2.3.2 Dispositifs matériels

Les interfaces les plus naturelles pour la tâche de navigation sont les dispositifs matériels demandant à l'utilisateur d'effectuer les mêmes mouvements que dans le monde réel, comme un tapis roulant pour une simple marche [Darken 97] ou un skate-board [Wang 12], illustrés sur la figure 2.11. Le Virtuix Omni™[Virtuix 13] est un dispositif grand public conçu pour que l'utilisateur se déplace librement (marche, course, ...) dans les jeux vidéos et autres applications de réalité virtuelle (voir la figure 2.12). Ce type de matériel améliore l'immersion dans la scène. Une interface permettant un mouvement naturel augmente le sensation de présence [Usoh 99], par exemple marcher réellement grâce à un tapis roulant en comparaison à marcher sur place.

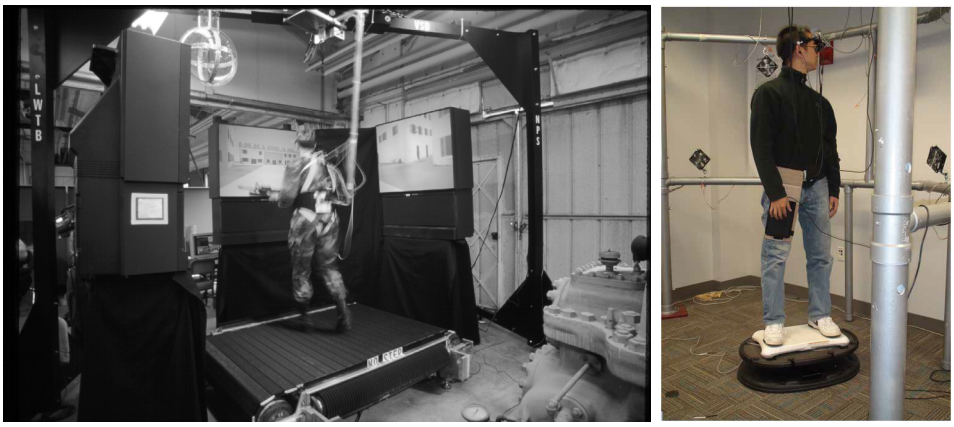


FIGURE 2.11 – Dispositifs matériels pour une navigation naturelle. **Gauche :** l'utilisateur se déplace à l'aide d'un tapis roulant, lui permettant de marcher dans le monde virtuel comme dans le monde physique. Image tirée de [Darken 97]. **Droite :** l'utilisateur utilise une Nintendo Wii Balance Board™comme un Hoverboard (skateboard volant futuriste) afin de naviguer dans les trois dimensions. Image tirée de [Wang 12].

Si les dispositifs matériels permettent une navigation naturelle, ils induisent également des problèmes de fatigue et de précision, et ne sont donc adaptés que pour des tâches d'exploration pure. Dans une tâche de recherche, ou pour une application où la navigation n'est qu'une étape préliminaire à la sélection et à la manipulation d'un objet, des techniques virtuelles sont souvent plus appropriées.

FIGURE 2.12 – Le Virtuix Omni^{TM} offre la possibilité de se déplacer librement dans le monde virtuel en marchant ou même en courant. Images tirées de [Virtuix 13].

2.3.3 Techniques virtuelles

De nombreuses techniques ont été mises au point pour permettre à un utilisateur de naviguer dans une scène. Les plus communes de ces techniques exploitent une représentation exocentrique de la scène, *i.e.* la scène vue d'un autre point de vue que celle de l'utilisateur. Une telle représentation peut être une vue 2D du dessus de la scène, ce qu'on appelle une minimap, ou un modèle réduit en 3D (*WIM*, *World-in-miniature*). Ces deux techniques sont illustrées en figure 2.13. L'utilisateur peut interagir avec ces représentations pour se déplacer instantanément, par exemple en sélectionnant une pièce d'un bâtiment afin de s'y « téléporter » directement. Néanmoins, cette téléportation peut perturber les utilisateurs en diminuant leur compréhension de leur orientation spatiale [Bowman 97]. Ces techniques sont principalement utilisées pour des tâches de recherche, dans le cas où l'utilisateur connaît la position de l'objet à atteindre.

FIGURE 2.13 – Deux exemples de techniques virtuelles exploitant une représentation exocentrique de la scène. **Gauche :** En haut à gauche de l'écran se trouve une vue de dessus de la scène. Image tirée de [Torres-Solis 09]. **Droite :** Une miniature $3D$ de la scène est placée au centre de l'écran. Image tirée de [Trueba Hornero 10].

2.4 Sélection

Dans la section 2.2, nous avons évoqué la tâche de manipulation. Manipuler un objet d'une scène virtuelle implique que l'utilisateur ait, d'une manière ou d'une autre, désigné cet objet au préalable. Certaines applications, en particulier concernant la visualisation de données scientifiques, se limitent à une scène ne contenant qu'un seul objet, comme une molécule agrandie. Les différentes manipulations disponibles sont implicitement appliquées à cet objet unique. Il n'est donc pas demandé à l'utilisateur de le sélectionner. De même, la sélection n'est pas pertinente dans les applications d'exploration, où seule une navigation dans le monde virtuel est disponible. Néanmoins, ce type d'applications reste rare. Pour reprendre nos exemples, il sera souvent possible pour l'utilisateur de manipuler plus finement la molécule, en supprimant ou déplaçant les atomes la composant, ou encore en combinant plusieurs molécules entre elles. De même, l'objectif de l'application n'est généralement pas d'explorer le monde virtuel, mais l'exploration est une étape nécessaire pour chercher un objet particulier et le manipuler. Dans ces deux situations, une technique de sélection est nécessaire.

La tâche de sélection en environnement immersif est le point central de notre travail. En conséquence, la partie II suivante est consacrée à cette tâche, et détaille plus précisément les enjeux et les solutions apportées à cette tâche. Dans le chapitre 3, nous présentons les différentes problématiques soulevées par la sélection, tandis que le chapitre 4 couvre un état de l'art des différentes techniques aidant l'utilisateur lors des différentes étapes qui composent cette tâche. Dans la partie III, nous présentons trois nouvelles solutions que nous avons développées pour répondre à des besoins précis dans notre contexte.

Ainsi, le chapitre 5 explore une nouvelle aide visuelle, le **Ring Concept** [Wonner 13], pour désigner dans quelle direction se trouve un objet particulier de la scène. Cette aide peut être aisément intégrée visuellement à d'autres outils, afin d'éviter un encombrement de la scène et de disperser l'attention de l'utilisateur. Le chapitre 6 détaille une technique de sélection, **Starfish** [Wonner 12b], guidant le pointeur vers sa cible à l'aide de contraintes virtuelles, créées par une surface implicite. Ces guides offrent à l'utilisateur une interaction plus confortable et plus rapide. Enfin, le chapitre 7 ouvre une nouvelle voie vers la prédiction de cibles en analysant le geste de sélection de l'utilisateur. Notre algorithme, **SPEED** [Wonner 11c], modélise le profil de vitesse du pointeur pour déterminer son point d'arrêt, où se trouve *a priori* sa cible.

2.5 Contexte

Dans le cadre de ce document, nous nous intéressons principalement à la tâche de sélection dans les environnements immersifs. Le contexte de notre travail porte sur la visualisation et la manipulation de modèles 3D complexes, et notamment les maillages volumiques, comme ceux représentés en figure 2.14. Les manipulations attendues pour ce type d'applications comprennent le déplacement, l'ajout, la suppression et l'édition des entités topologiques du maillage : sommet, arête, face et volume. Or, comme nous l'avons vu, la manipulation de telles entités est précédée d'une étape de sélection, qui est le cœur de notre travail. Cette étape peut être difficile, notamment à cause de problèmes de densités et d'occlusions de la scène. Notre travail consiste alors dans un premier temps à décrire les différentes problématiques liées à cette tâche de sélection et les solutions existantes. Dans un second temps, nous pointons les limitations de ces solutions avant de proposer nos propres méthodes et techniques répondant à des problématiques précises.

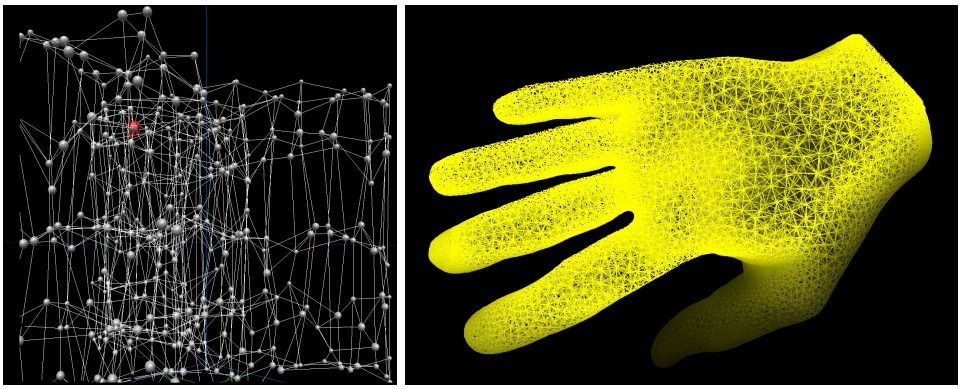


FIGURE 2.14 – Exemples de maillages volumiques. L'utilisateur peut être amené à interagir avec ces maillages, par exemple en déplaçant ou supprimant des sommets. La sélection d'un tel sommet est une étape préliminaire à cette interaction.

2.6 Conclusion

L'objectif de ce chapitre était de distinguer les types de tâche que pouvait réaliser un utilisateur dans une scène virtuelle, et de présenter les moyens à sa disposition pour les effectuer. Toutes les applications de réalité virtuelle ne permettent pas nécessairement de réaliser les quatre tâches. Par exemple, une application dont l'objectif est d'offrir une visualisation 3D d'une molécule d'ADN [Hérisson 04], ou d'une pièce réalisée en CARV (*Conception Assisté par Réalité Virtuelle*) [Bourdot 10, Convard 04] ne requiert pas de naviguer dans la scène, mais principalement de manipuler l'objet affiché. À l'inverse, dans le domaine médical, les thérapies basées sur l'exposition sous réalité virtuelle (ERV) exposent le patient à des environnements anxiogènes, comme un pont suspendu pour une personne acrophobe [Rothbaum 95], ou une pièce contenant des araignées pour un arachnophobe [Carlin 97, Garcia-Palacios 02]. Dans la plupart des cas, ces applications exposent le patient à une situation particulière pour l'accoutumer, et ne demandent pas d'interaction avec les objets de la scène.

Dans cette première partie, nous avons offert un aperçu général de la réalité virtuelle, de ses applications, et des interactions possibles que l'utilisateur peut effectuer dans des environnements de réalité virtuelle. La partie II suivante est consacrée aux problématiques liées à la sélection, et aux techniques qui ont été développées dans la littérature pour répondre à ces problématiques.

Deuxième partie

Sélection : théorie et état de l'art

Chapitre 3

Contexte théorique

I've got a theory. It could be bunnies !

Anya Jenkins,
Buffy the Vampire Slayer
Episode 6x07, *Once More, with Feelings*

Sommaire

3.1 Problématiques de la sélection 37
3.1.1 Loi de Fitts 37
3.1.2 Stéréoscopie 38
3.1.3 Densité et occlusions 39
3.1.4 Tremblements naturels 39
3.1.5 Bilan 39
3.2 Phases de la sélection 40
3.2.1 Phase de décision 40
3.2.2 Phase balistique 41
3.2.3 Phase de contrôle 41
3.3 Conclusion 41

La majorité des systèmes d'exploitation sur stations de travail utilisent le paradigme *WIMP* (*Windows, Icons, Menus and Pointing Devices*). La sélection d'un élément dans ces environnements, comme une fenêtre, une icône ou un menu, est effectuée à l'aide d'un pointeur contrôlé par l'utilisateur par l'intermédiaire d'une souris. Le mouvement du pointeur est donc contraint dans un plan 2D. Néanmoins, l'exploration et la navigation dans une scène 3D sont possibles à l'aide d'une molette sur la souris, ou en utilisant un second dispositif, comme le Globefish [Kulik 09] ou certaines touches du clavier.

Dans une application en environnement de réalité virtuelle, l'utilisateur manipule un dispositif pouvant être déplacé dans les trois dimensions. L'interaction 3D est donc naturelle et permet une sélection et une manipulation directe de l'objet dans le monde virtuel [Shneiderman 97]. Néanmoins, l'ajout de cette troisième dimension pose un certain nombre de problèmes, comme les occlusions entre les objets de la scène. Dans ce chapitre, nous détaillons dans un premier temps les problématiques liées à la sélection d'objets dans un environnement de réalité virtuelle. Dans un second temps, nous examinons les différentes étapes lors de la sélection d'un objet.

3.1 Problématiques de la sélection

Nous nous intéressons ici aux différents problèmes engendrés par l'interaction en environnement de réalité virtuelle, et en particulier par la sélection. Les origines de ces problèmes sont diverses [Hinckley 94] : certaines sont liées à la géométrie de la scène et des objets virtuels, d'autres sont dues à des contraintes techniques ou humaines.

3.1.1 Loi de Fitts

Intuitivement, nous pouvons affirmer qu'un objet de grande taille et proche de l'utilisateur est plus facile à sélectionner qu'un objet de petite taille et loin de l'utilisateur. Cette observation est également valable sur station de travail classique : nous pouvons naturellement supposer que le temps nécessaire pour sélectionner une entité comme une icône dépend de sa taille et de sa distance avec le pointeur de l'utilisateur.

En 1954, Paul M. Fitts [Fitts 54] mène une expérience dans laquelle l'utilisateur doit atteindre alternativement le centre de deux cibles avec un stylet, comme illustré en figure 3.1. Les résultats de cette expérience conduisent Fitts à proposer une loi donnant le temps nécessaire pour sélectionner une cible de largeur W située à une distance D :

$$T = a + b \times \log_2 \left(\frac{2D}{W} \right), \tag{3.1}$$

où a et b sont des constantes déterminées empiriquement en fonction de l'utilisateur et de l'environnement utilisé. Le terme logarithmique est appelé l'indice de difficulté, noté I_d, de la cible. Dans les expériences de Fitts, le dispositif de pointage et les cibles à atteindre sont physiques. Cependant, le pointage d'une cible dans un monde virtuel suit également la loi de Fitts [Card 78, Wingrave 05].

Actuellement, la forme communément utilisée de la loi de Fitts est la formulation de Shannon [MacKenzie 92b], où l'indice de difficulté est remplacé par le terme suivant :

$$I_d = \log_2 \left(1 + \frac{D}{W} \right) \tag{3.2}$$

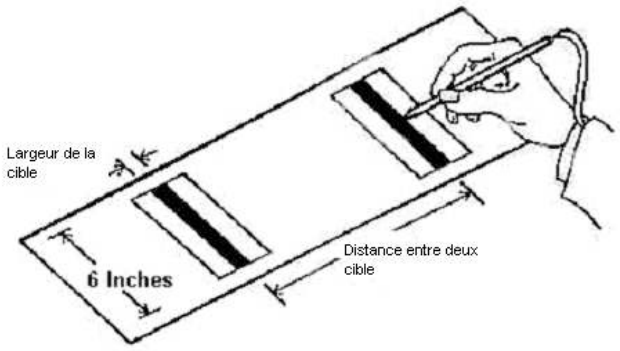


FIGURE 3.1 – Expérience de « va-et-vient » conduite par Paul Fitts [Fitts 54]. Le temps nécessaire pour atteindre une cible dépend de sa taille et de sa distance.

De nombreuses études ont étendu la loi de Fitts pour des tâches de pointage en deux dimensions [MacKenzie 92a, Murata 99, Accot 03], dans des espaces 3D [Grossman 04, Teather 11, Argelaguet 09, Kopper 10, Iwase 01], lors de sélections successives [Huys 10], ou encore pour tenir compte du bruit [Monk 85], de la latence [Ware 94], ou de la densité de la scène [Blanch 11].

Dans le contexte de notre travail, nous retenons de ces études que le temps requis pour atteindre un objet décroît avec la taille de l'objet, et croît avec sa distance avec l'utilisateur. Une technique souhaitant améliorer le temps de sélection pourrait agir sur la distance et la taille relative des objets. Le système de *Dock*, initialement apparu dans l'environnement graphique Mac OS X, applique ce principe en agrandissant les icônes proches du pointeur de l'utilisateur, facilitant leur sélection [McGuffin 05].

3.1.2 Stéréoscopie

Le problème qui se présente immédiatement lorsque nous regardons une scène 3D sur un écran 2D est celui de la distance relative entre les objets. Nous percevons difficilement la profondeur et la taille des objets, car nous n'observons alors qu'une projection 2D de la scène. Un objet petit et proche peut être interprété comme un objet grand mais lointain. L'utilisation de la stéréoscopie, très courante en réalité virtuelle, ajoute une dimension supplémentaire à la perception de la scène, et permet de mieux distinguer la profondeur relative des objets [Cutting 97]. Néanmoins, un utilisateur avec un dispositif stéréoscopique perçoit les distances dans l'espace virtuel plus petites qu'elles ne le sont réellement, d'environ 70% [Sahm 05, Willemsen 08]. L'origine de ce phénomène n'est pas clairement établie, même s'il semble que les distances ne soient pas compressées si le monde virtuel reproduit exactement une scène bien connue de l'utilisateur [Interrante 06]. De plus, la profondeur reste la dimension la plus difficile à appréhender, comme le montre une étude lors d'une tâche de sélection [Fiorentino 03]. L'erreur commise en pointant précisément

un point de l'espace est plus élevée (d'un facteur 2 environ) dans la dimension de la profondeur que dans les dimensions verticale et horizontale.

De plus, l'utilisation prolongée de la stéréoscopie peut provoquer une fatigue visuelle et des maux de tête. L'origine de ces troubles est l'objet de plusieurs études [Liu 09b, Kim 11]. Cependant, ces problèmes n'affectent qu'un faible pourcentage de la population, et la vision stéréoscopique reste largement employée. Des aides visuelles sont alors couramment employées afin de percevoir la profondeur relative des objets, par exemple en affichant leurs ombres.

3.1.3 Densité et occlusions

Nous avons vu que la loi de Fitts permettait d'estimer le temps requis pour sélectionner une cible isolée en fonction de sa taille et de sa distance. Or, dans une application 2D, par exemple une fenêtre contenant plusieurs icônes, le cas d'un objet isolé n'est pas réaliste. Des études [Blanch 11] montrent que les objets autour de la cible, les *distracteurs*, ont un impact sur le temps requis pour sélectionner cette cible, et proposent de rajouter un facteur de densité à la loi de Fitts.

Dans une scène 3D, le problème de densité s'accompagne du phénomène d'occlusion entre les objets de la scène. Un distracteur proche de l'utilisateur peut occulter partiellement ou entièrement un autre objet, modifiant la perception de la scène de l'utilisateur, et impactant les performances de l'utilisateur [Vanacken 07]. Certaines techniques [Carpendale 97, Elmqvist 05] déforment l'espace afin de réduire les occlusions, mais ces solutions sont indépendantes du mécanisme de sélection.

3.1.4 Tremblements naturels

Les problèmes cités précédemment impliquent que la sélection d'une cible nécessite souvent une grande concentration et une grande précision de la part de l'utilisateur. Or, les mains de cet utilisateur sont sujettes à des tremblements involontaires mais naturels, engendrés par l'absence de support physique. Contrairement à une station de travail classique où la main de l'utilisateur repose sur un dispositif comme la souris et est contrainte sur un plan, l'interaction 3D inférée par un environnement de réalité virtuelle requiert une main non contrainte dans les trois dimensions. Dans ces conditions, l'amplitude de ces tremblements naturels augmente.

Un phénomène analogue, le bruit, est également observé sur les dispositifs d'interaction [Teather 09]. Même si l'un de ces dispositifs est immobilisé, des fluctuations dans les positions enregistrées peuvent être notées. Les tremblements naturels amplifient ce bruit. Les deux phénomènes troublent la précision de l'utilisateur.

3.1.5 Bilan

Nous retenons de ces observations que la conception d'une technique améliorant les performances d'un utilisateur lors d'une tâche de sélection devrait aborder les problèmes que nous avons cités. Par exemple, contraindre les mouvements de la main permettrait de compenser les tremblements involontaires induits par l'absence de support physique, et donc amènerait à une augmentation de la précision de l'utilisateur.

Dans la section suivante, nous abordons les approches que nous souhaitons développer pour concevoir des techniques d'aides à la sélection en environnement de réalité virtuelle.

3.2 Phases de la sélection

Nous nous intéressons ici au cas où l'utilisateur manipule un pointeur dans l'espace, et où la sélection s'effectue en déplaçant ce pointeur sur l'objet à sélectionner. Nous décrivons d'autres types de sélection en section 4.4.

Nous décrivons dans cette section ce que nous considérons comme les phases importantes d'une sélection, qu'il s'agisse d'une sélection dans une scène en deux ou trois dimensions. Chacune de ces phases pose des problématiques différentes auxquelles nous cherchons à proposer des solutions. Nous décomposons le geste en trois phases : phase de décision, phase balistique et phase de contrôle. Cette approche est inspirée de différentes études [Gordon 94, Huegel 09]. En particulier, le *two-component model* de Woodworth [Woodworth 99, Elliott 01] stipule qu'un geste pour atteindre une cible est découpé en une impulsion initiale afin de s'approcher du voisinage de cette cible, suivie d'une série de corrections de petits mouvements. Le modèle que nous proposons ici ajoute une phase de décision durant laquelle l'utilisateur choisit sa cible. Nous supposons également que l'utilisateur, dans son impulsion initiale, commence déjà ses corrections de trajectoire et de vitesse une fois atteint son pic de vélocité.

3.2.1 Phase de décision

Avant d'entamer un geste de sélection, il est nécessaire de localiser la cible. L'utilisateur observe la scène, repère sa cible et estime le mouvement qu'il doit effectuer pour l'atteindre. Durant cette phase, l'utilisateur est presque immobile (il ne peut l'être parfaitement à cause de l'absence de support physique). Une mauvaise connaissance de la disposition de la scène, du fait des occlusions entre objets ou d'une mauvaise perception de la profondeur, peut perturber la décision de l'utilisateur et fausser l'amplitude et la trajectoire de son impulsion initiale. Localiser correctement la position de la cible est donc une étape préliminaire indispensable au geste de sélection.

De nombreuses applications fournissent ainsi des outils pour aider l'utilisateur à repérer la position des objets d'intérêts dans la scène par rapport à sa propre position, comme un radar ou une flèche 3D [Chittaro 04]. Même pour des objets visibles à l'écran, ces aides visuelles permettent une meilleure appréhension de la direction à emprunter pour les atteindre. Nous verrons dans la section 4.1 un état de l'art de ces techniques.

3.2.2 Phase balistique

Lorsque l'utilisateur a décidé quelle cible il souhaitait atteindre, il entame son geste de sélection par une impulsion initiale, que nous appelons phase balistique. Sa vitesse augmente pour parcourir rapidement la distance qui le sépare de sa cible, et arriver dans son voisinage. Il est montré que la vitesse maximale atteinte durant cette impulsion croît avec la distance de la cible. Ce pic de vitesse peut ainsi être utilisé pour estimer la position de la cible avant que l'utilisateur ne l'ait atteinte, par exemple en supposant une relation linéaire entre le pic de vitesse et la distance à effectuer [Asano 05]. Cette information peut alors être exploitée en sélectionnant automatiquement la cible correspondante, ou en amenant directement le pointeur de l'utilisateur à l'endroit visé. L'objectif de cette manœuvre est d'améliorer les performances de l'utilisateur en lui évitant d'effectuer la totalité de son geste de sélection. Plusieurs algorithmes, appelés algorithmes de prédiction, et que nous décrivons en section 4.2, analysent le profil de vitesse des mouvements de l'utilisateur pour prédire leur cible.

3.2.3 Phase de contrôle

Après la phase balistique, l'utilisateur, dont le pointeur est arrivé dans le voisinage de sa cible, corrige sa trajectoire et sa vitesse durant la phase de réajustement (ou phase de contrôle). Durant cette phase, l'utilisateur essaye d'être précis et se déplace à une vitesse peu élevée. Il s'agit généralement de la phase la plus longue, car elle demande une grande précision et donc de nombreuses corrections de trajectoires. Une fois encore, la mauvaise perception de la scène, les occlusions entre objets et les tremblements involontaires de la main entravent l'utilisateur dans sa tâche. Différentes techniques ont été développées afin d'éviter une perte de temps et de confort pour l'utilisateur. Par exemple, le Bubble Cursor [Grossman 05] permet de sélectionner une cible dès que le pointeur se trouve dans son voisinage. Nous décrivons ces techniques dans la section 4.3.

3.3 Conclusion

Dans ce chapitre, nous avons présenté les principales problématiques liées à la sélection en environnement immersif. Nous avons également décomposé le geste de sélection en trois phases : phase de décision, phase balistique, et phase de contrôle. Cette décomposition nous permet d'identifier et de regrouper les différentes catégories de techniques développées pour aider l'utilisateur à sélectionner sa cible, et pour améliorer ses performances.

Dans le chapitre 4 suivant, nous commençons par un état de l'art de ces différentes techniques selon leur apport à la tâche de sélection. L'étude de leurs avantages et inconvénients respectifs nous conduit à nous interroger sur la conception de nouvelles approches plus performantes et confortables pour l'utilisateur. Nous présentons brièvement à la fin du chapitre suivant les trois nouvelles techniques que nous avons développées dans ce cadre, et qui sont l'objet de la partie III.

Chapitre 4

État de l'art

> The ancient teachers of this science promised impossibilities and performed nothing. The modern masters promise very little [but] have indeed performed miracles.
>
> Mary Wollstonecraft Shelley,
> *Frankenstein*

Sommaire

4.1 Localisation 45
4.1.1 Techniques exocentriques 45
4.1.2 Techniques égocentriques 48
4.2 Prédiction 52
4.3 Avatar de la main 55
4.4 Rayons 60
4.5 Aides proposées 66
4.5.1 Localisation de cible 66
4.5.2 Avatar de la main 67
4.5.3 Prédiction de fin de mouvement 67
4.6 Conclusion 68

L'interaction en réalité virtuelle est un domaine très étudié qui a généré de nombreux algorithmes et techniques, en particulier pour améliorer les performances de l'utilisateur. Dans le cadre de la tâche de sélection, plusieurs approches sont abordées. Certaines méthodes s'attachent à la localisation des objets de la scène, permettant à l'utilisateur de planifier son geste de sélection, en particulier dans des environnements denses. D'autres algorithmes déduisent du mouvement de l'utilisateur la cible visée par ce mouvement,

permettant ainsi de raccourcir, voire de supprimer, la phase de contrôle du geste de sélection. Enfin, des techniques d'interaction sont développées pour faciliter et accélérer la sélection elle-même. La grande majorité de ces techniques peut être divisée en deux catégories [Mine 95, Bowman 99b, Dang 07], d'une part les techniques basées sur un rayon virtuel lancé depuis l'utilisateur et traversant la scène, et d'autre part les techniques basées sur la manipulation d'un avatar de la main de l'utilisateur.

Nous présentons dans ce chapitre un état de l'art de plusieurs aspects de la sélection en réalité virtuelle. Nous nous intéressons dans un premier temps aux techniques de localisation des entités dans l'environnement virtuel. Dans un second temps, nous décrivons les méthodes analysant le geste de sélection de l'utilisateur pour prédire la cible visée. Nous détaillons ensuite les techniques de sélection basées sur un avatar de la main de l'utilisateur, puis les techniques basées sur un rayon virtuel. Enfin, nous décrivons les approches que nous souhaitons aborder comme solution aux problématiques de la sélection. Ces approches seront détaillées plus en avant dans la partie III.

4.1 Localisation

Localiser un objet de la scène virtuelle et se représenter mentalement sa position relative à l'utilisateur sont deux étapes préliminaires à la sélection de cet objet. Parmi les techniques existantes permettant de localiser un objet dans la scène, nous distinguons les approches **égocentriques**, *i.e.* celles offrant une vision de la scène selon la même perspective que l'utilisateur, des approches **exocentriques**, *i.e.* celles dont la perspective de la scène est différente de celle de l'utilisateur. Cette distinction provient de la classification proposée par Milgram et Kishino [Milgram 94], puis par Bowman *et al.* [Bowman 99b].

4.1.1 Techniques exocentriques

Une approche courante est d'utiliser un *radar* 2D [Tonnis 05] affichant la scène virtuelle vue de haut, comme le montre la figure 4.1. Le radar, souvent utilisé dans les jeux vidéo (voir la figure 4.2), affiche une vue globale de la scène, ou une portion locale centrée sur l'utilisateur, comme Bird's Eye [Tonnis 06]. Dans les environnements immersifs, cette technique est particulièrement intéressante pour visualiser les objets non visibles car éloignés de l'utilisateur ou derrière lui, ou encore parce qu'ils sont cachés par d'autres objets. Ruddle *et al.* [Ruddle 99] proposent d'utiliser simultanément un radar local et un radar global afin de profiter des avantages des deux visualisations.

Bien que le radar permet d'afficher plusieurs points d'intérêt de la scène, une rotation mentale entre la perspective exocentrique du radar et la perspective égocentrique de l'utilisateur, est requise pour inférer de cette visualisation la position de l'objet dans la scène. Il est montré que la surcharge mentale induite par ces rotations augmente avec le nombre d'objets dans la scène [Aretz 92, Darken 99]. De plus, ce type de technique ne permet pas de distinguer des objets ayant les mêmes abscisses et ordonnées mais placés

à différentes hauteurs, et n'est dont pas adaptée pour les scènes 3D [Chittaro 04].

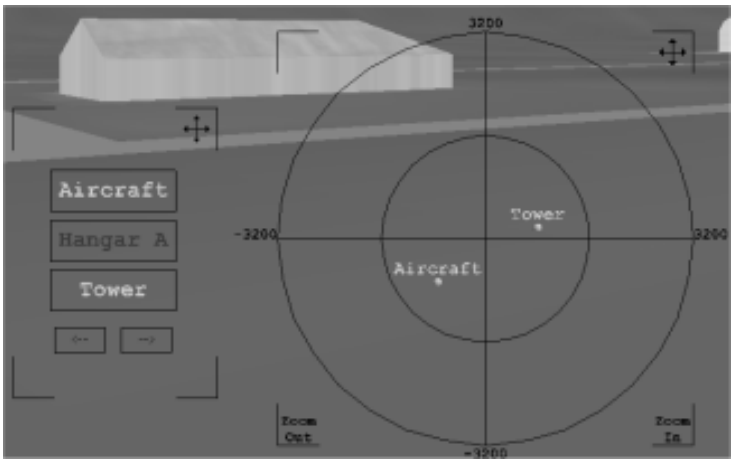


FIGURE 4.1 – Aperçu d'un radar 2D. Chaque objet d'intérêt est représenté par un point sur le radar. Image tirée de [Chittaro 04].

FIGURE 4.2 – Utilisation du radar dans les jeux vidéo. Les catégories d'objets dans la scène sont différenciées par des icônes différentes. **À gauche :** Le radar affiche la scène complète. Image extraite du jeu *Warcraft III : Reign of Chaos*, de Blizzard Entertainment®. **À droite :** Le radar n'affiche qu'une petite partie de la scène, centrée sur l'avatar du joueur représenté par une flèche verte. Image extraite du jeu *Battlefield 4* de Electronic Arts™.

Pour surmonter les limitations des radars, une technique appelée *World In Miniature* (WIM) a été proposée [Stoakley 95, Coffey 11]. Un WIM est une version 3D de la scène en plus petit, comme illustré en figure 4.3. De la même manière qu'un radar, un WIM peut afficher la scène dans son intégralité ou seulement une partie. L'utilisateur peut

interagir simultanément avec le WIM ou avec l'environnement virtuel. Bien que cette solution permet, à l'inverse des radars, de localiser des objets dans les trois dimensions, les WIMs surchargent généralement la scène et diminuent l'immersion et l'interaction avec l'environnement. De plus, de même que pour les radars, le changement permanent de système de coordonnées entraîne une surcharge mentale pour l'utilisateur.

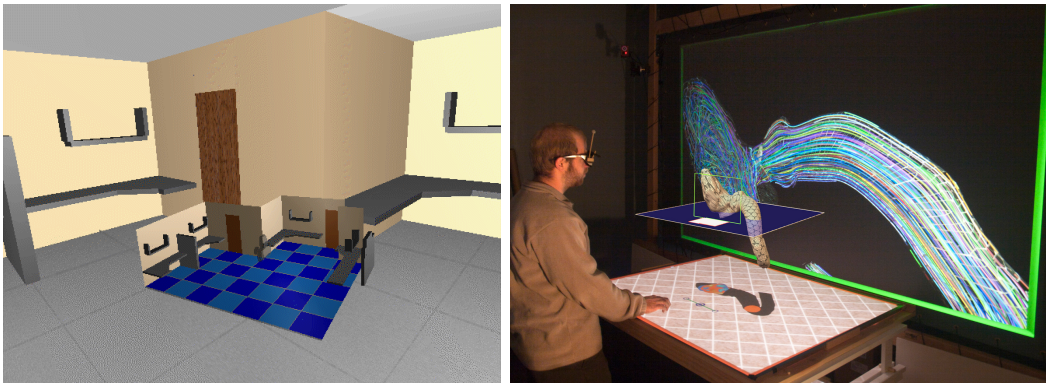


FIGURE 4.3 – Aperçu des World In Miniature. La version miniature du monde s'affiche devant l'utilisateur. **À gauche :** WIM dans une application de navigation. Image tirée de [Stoakley 95]. **À droite :** WIM dans une application de visualisation de donnée. Image tirée de [Coffey 11].

Flower Garden [McCrae 10] est un WIM affichant tous les objets dans une version plane de la scène, comme l'illustre la figure 4.4. Chaque objet est placé en haut d'une tige d'autant plus longue que sa distance à l'utilisateur est grande. Dans cette visualisation, deux objets sont adjacents si le changement d'orientation de caméra pour passer de l'un à l'autre est faible. Si le *Flower Garden* permet de visualiser tous les objets en modifiant la disposition de la scène, et donc en diminuant *a priori* le phénomène d'occlusion, cette solution n'indique pas directement la position des objets relativement à l'utilisateur, et ne permet donc pas de localiser précisément et surtout rapidement un objet dans l'espace virtuel.

De cette étude, nous retenons que les techniques exocentriques sont adaptées pour localiser simultanément tous les objets dans l'environnement et par rapport à l'utilisateur. Néanmoins, si elles permettent d'avoir une conscience générale de la scène, elles n'apportent généralement pas d'aide localement [Wang 04]. De plus, ces techniques affichent la scène avec une autre perspective que celle de l'utilisateur, impliquant qu'une rotation mentale est nécessaire avant de planifier un éventuel mouvement de sélection. À l'inverse, les techniques dites **égocentriques**, auxquelles nous allons maintenant nous intéresser, donnent une indication sur la position de l'objet directement dans l'espace virtuel, l'indice visuel étant donné dans la perspective de l'utilisateur.

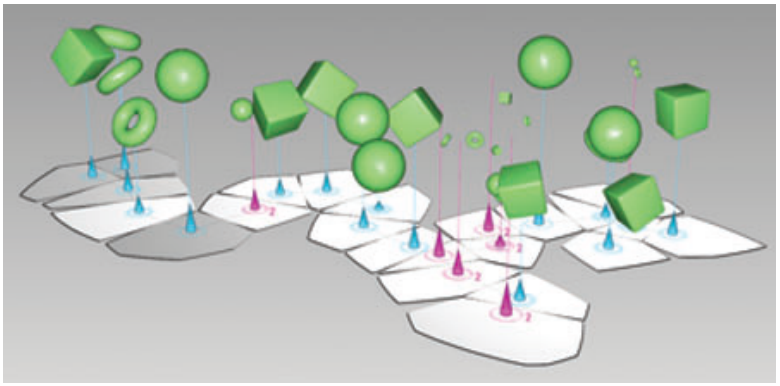


FIGURE 4.4 – Aperçu du Flower Garden. La hauteur des tiges indique la distance de l'objet à l'utilisateur. Image tirée de [McCrae 10].

4.1.2 Techniques égocentriques

Lorsque l'utilisateur recherche un objet spécifique dans l'espace virtuel, cet objet peut être directement visible du point de vue de l'utilisateur. À l'inverse, si l'objet est trop loin de l'utilisateur, ou derrière lui, ou encore s'il est caché par un distracteur, l'utilisateur ne peut le voir et entamer la planification du geste de sélection qu'en se déplaçant. Nous ne nous intéresserons pas ici aux différentes méthodes pour mettre en évidence un objet déjà visible, mais aux techniques mises en œuvre pour indiquer dans quelle direction se trouve un objet non visible.

Dans ce but, la technique la plus intuitive et la plus couramment employée est l'utilisation d'une flèche 3D. Cette flèche, le plus souvent placée en face de l'utilisateur, est orientée vers les points d'intérêt de la scène, comme l'illustre la figure 4.5. Pour une tâche de navigation dans un environnement où l'utilisateur ne peut que marcher sur un plan, l'utilisation d'une flèche est aussi performante qu'un radar 2D. Dans une scène où l'utilisateur peut voler dans les trois dimensions, la flèche semble surpasser le radar, principalement en raison du fait que le radar ne permet pas aisément d'indiquer la hauteur des objets [Chittaro 04].

Néanmoins, la direction d'une simple flèche 3D, principalement basée sur un cône et un cylindre, peut être difficile à interpréter, en particulier lorsque la flèche pointe directement vers l'utilisateur ou dans la direction opposée. Des ailettes permettent de lever cette ambiguïté [Tonnis 06] (voir la figure 4.6).

Dans une scène virtuelle, il est souvent intéressant d'indiquer la position de plusieurs points d'intérêt. L'extension logique de la flèche est évidemment le groupe de flèches. Néanmoins, cette solution présente deux difficultés. D'une part, si plusieurs objets sont pointés, une indication supplémentaire semble nécessaire pour les différencier, comme

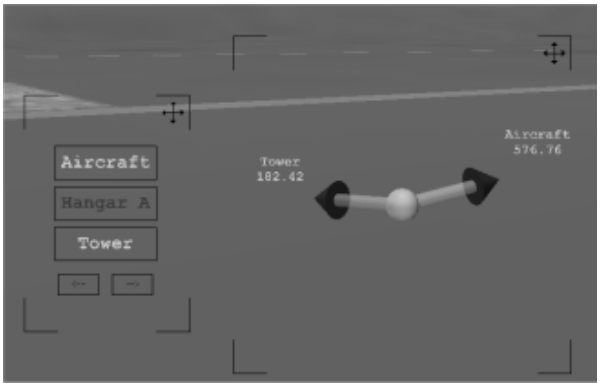


FIGURE 4.5 – Aperçu de deux flèches 3D pointant vers deux lieux de la scène que l'utilisateur doit atteindre. L'identité et la distance de ces lieux sont ajoutées à coté de chaque flèche. Image tirée de [Chittaro 04].

FIGURE 4.6 – Aperçu de deux flèches 3D dont l'indication de direction est renforcée par l'ajout d'ailettes. Image tirée de [Tonnis 06].

indiquer textuellement le nom de l'objet [Chittaro 04] (voir la figure 4.5), ou en affichant une miniature de l'objet comme le *Mirror Ball* [McCrae 10] (voir la figure 4.7). D'autre part, un nombre de flèches élevé entraîne une occlusion importante de la scène, voire même des inter-occlusions au sein même du groupe de flèches. Ceci est particulièrement visible sur la figure 4.7. La technique appelée *Wedge Sphere* réduit cette occlusion en ne conservant qu'un cône pour chaque flèche, en supprimant le cylindre. Toutefois, les ambiguïtés dans l'interprétation des informations de direction semblent plus marquées que pour une simple flèche 3D.

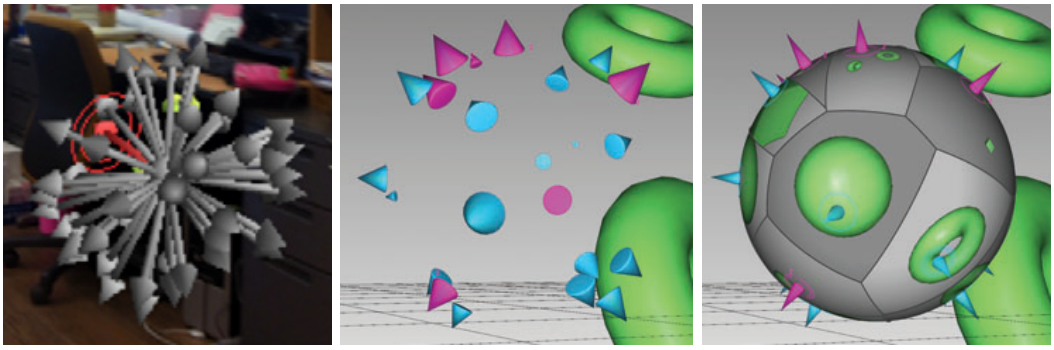


FIGURE 4.7 – Groupes de flèches. **Gauche :** Un groupe de 50 flèches. Les occlusions de la scène et entre les flèches sont mises en évidence [Jo 11]. **Milieu :** *Wedge Sphere* [McCrae 10]. Chaque objet est uniquement représenté par un cône. **Droite :** *Mirror Ball* [McCrae 10]. Une miniature des objets est affichée pour identifier les flèches.

Pour résoudre ce problème d'occlusion, certaines techniques occupent moins d'espace visuel en n'affichant que quelques indices sur la position des objets non visibles. *Aroundplot* [Jo 11] affiche un point sur le bord de l'écran pour indiquer dans quelle direction l'utilisateur doit se tourner pour apercevoir l'objet, comme l'illustre la figure 4.8. En exploitant ce concept, *City Lights* [Zellweger 03] affiche des points pour indiquer la direction d'un objet, des segments pour indiquer la taille de cet objet, ou des arcs de cercle pour inférer directement sa position. Cette technique est illustrée en figure 4.9. *Wedge* [Gustafson 08] adapte ces arcs de cercles en évitant les chevauchements, facilitant la lisibilité des informations (voir figure 4.10). Toutes ces solutions sont particulièrement adaptées au domaine de la réalité augmentée, où les écrans utilisés sont généralement de petites tailles (téléphones, tablettes, lunettes...).

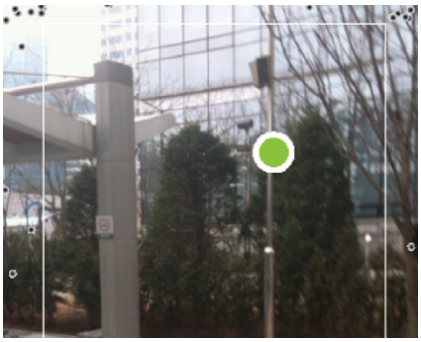


FIGURE 4.8 – Technique *Aroundplot* [Jo 11]. Un point noir sur le bord de l'écran représente un objet non visible, et indique dans quelle direction il faut tourner l'écran pour que l'objet soit visible. Les points verts sont les objets déjà visibles à l'écran.

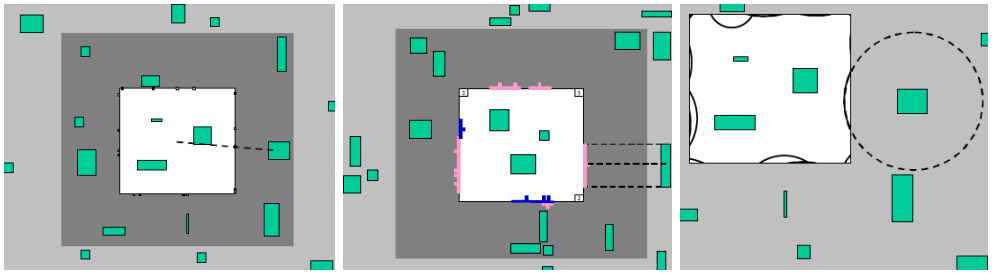


FIGURE 4.9 – Technique *City Lights* [Zellweger 03]. À gauche, un point indique la direction de l'objet. Au milieu, un segment figure la projection de l'objet sur le bord de l'écran. À droite, un arc de cercle permet d'inférer la position de l'objet. Le choix de l'une ou l'autre de ces indices visuels dépend du contexte applicatif.

FIGURE 4.10 – Aperçu de la technique *Wedge* [Gustafson 08]. À gauche, les indications de direction sont données par les arcs de cercle de la technique *City Lights*. Des chevauchements sont visibles, et gênent la lisibilité. À droite, les mêmes informations sont données par la technique *Wedge*. L'algorithme adapte la taille des indices visuels pour empêcher les chevauchements.

L'étude que nous venons de mener présente un état de l'art des méthodes employées afin de localiser une ou plusieurs entités dans la scène virtuelle. Dans la grande majorité des applications de réalité virtuelle, localiser une entité n'est pas une fin en soi, mais une étape nécessaire avant de naviguer vers elle et la sélectionner, afin de la manipuler. Dans le cadre de notre travail, nous nous intéressons plus particulièrement à la sélection de ces entités, et donc à une aide locale pour repérer la position des entités à sélectionner. Des travaux antérieurs ont montré que les techniques égocentriques fournissaient une aide locale plus performante que les techniques exocentriques [Wang 04]. Nous verrons en section 4.5 l'élaboration d'une nouvelle technique égocentrique proposant une alternative aux flèches 3D. Cette technique sera décrite plus en avant dans le chapitre 5.

Dans la section suivante, nous nous intéressons au geste de sélection, et en particulier aux méthodes analysant ce geste afin d'en déduire la cible visée.

4.2 Prédiction

Comme nous l'avons vu au chapitre 3, un geste de sélection peut se décomposer en une phase d'approche rapide mais peu précise, et une phase de contrôle où l'utilisateur ajuste son mouvement pour atteindre précisément sa cible [Woodworth 99, Elliott 01]. Analyser le geste de sélection afin de déterminer, ou « prédire », sa cible, permet de réduire cette dernière phase, plus lente, en sélectionnant automatiquement l'entité avant que l'utilisateur ne l'atteigne. Néanmoins, dans le cas de scènes denses, il est vain de vouloir prédire l'entité visée sans erreur. On cherche alors à délimiter une zone dans laquelle elle se trouve « presque sûrement ». Cette information peut alors être utilisée pour faciliter la phase de contrôle, par exemple en agrandissant les objets dans cette zone [Ruiz 10]. Les systèmes haptiques, nécessitant de nombreux calculs afin de simuler au mieux l'interaction entre l'utilisateur et les objets de la scène, exploitent également ces prédictions afin d'optimiser ces calculs [Hasegawa 06].

Le premier algorithme de prédiction a été proposé en 1998 par Murata [Murata 98], pour une application 2D sur une station de travail classique. Son approche compare deux positions successives du curseur de la souris pour définir une direction de mouvement. L'entité qui minimise l'écart angulaire avec cette direction de mouvement, pendant un certain nombre d'intervalles de temps, est la cible prédite par l'algorithme. Il est néanmoins clair que cette méthode n'est pas adaptée dans la plupart des scènes, où plusieurs entités peuvent être alignées avec la direction du mouvement, car elle ne permet pas de désambiguïser ces entités.

Si la trajectoire du mouvement de sélection reste un indice important pour déterminer dans quelle direction se trouve la cible, d'autres études s'intéressent au profil de vitesse de ce mouvement pour en déduire à quelle distance cette cible est placée. Ce profil de vitesse, illustré en figure 4.11, présente un pic de vélocité. Le *Delphian Desktop* émet l'hypothèse d'une relation linéaire entre l'amplitude de ce pic et la distance de la

cible [Asano 05]. Lorsque le pic de vélocité est détecté, une distance est inférée de son amplitude. La trajectoire courante du curseur de la souris et cette distance donnent une position dans le plan, sur laquelle le curseur est envoyé. Les résultats de l'expérience menée par les auteurs montrent un gain de temps pour atteindre les cibles lointaines, mais également une perte d'efficacité lorsque la cible est proche. Cette méthode a été reprise dans un environnement $3D$ et haptique à l'aide d'un Phantom [Wonner 10a], mais la taille réduite de la scène utilisée et de l'espace de travail fourni par ce périphérique n'a pas permis de conclure à l'efficacité de cette méthode.

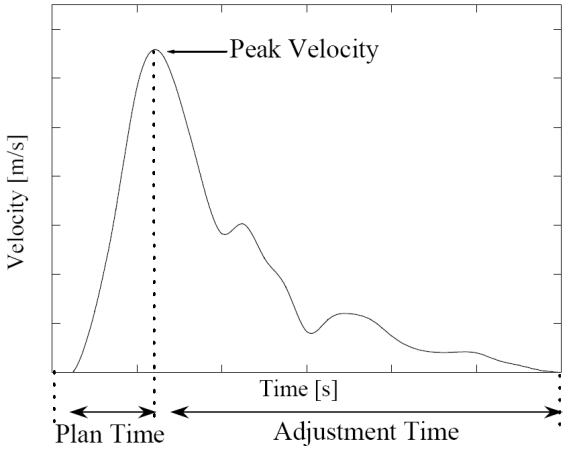


FIGURE 4.11 – Profil de vitesse pendant un tâche de sélection. Image tirée de [Asano 05].

Se baser uniquement sur le pic de vélocité ne semble donc pas suffisant. Hogan observe que la trajectoire d'un geste de sélection chez les primates minimise les variations du jerk [1] et propose une modélisation de la trajectoire de ce geste par une équation polynomiale de degré 5 [Hogan 84]. Ce principe, le *minimum jerk law* [Richardson 02], est repris et validé par Huegel *et al.* pour des mouvements humains [Huegel 09]. L'algorithme KEP, pour *Kinematic Endpoint Prediction* [Lank 07, Ruiz 09], exploite ce principe en modélisant à chaque instant le profil de vitesse (courbe de vitesse en fonction de la distance parcourue) par une fonction quadratique, comme illustré en figure 4.12. La fonction obtenue a deux racines, indiquant donc la distance à parcourir pour que la vitesse s'annule, et donc pour atteindre la cible. Le mouvement étant considéré rectiligne, la position de la cible est facilement déduite. Cependant, comme nous l'avons vu, la phase balistique, sur laquelle se base principalement KEP, est peu précise et conduit donc à des mauvaises prédictions.

Dans le domaine des interfaces cerveau-ordinateur [2], l'activité neuronale est analysée et certaines caractéristiques sont comparées à un « vocabulaire cérébral » cor-

1. Le jerk est la dérivée troisième de la position par rapport au temps.
2. BCI : *Brain Computer Interfaces.*

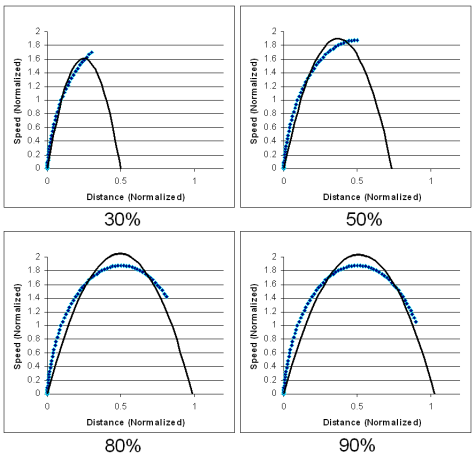


FIGURE 4.12 – Modélisation du profil de vitesse à plusieurs instants. Image tirée de [Lank 07].

respondant généralement à des actions enregistrées au préalable. Ces interfaces sont utilisées dans certaines applications de réalité virtuelle comme outils d'interaction, en particulier par le projet OpenViBE (*Open Platform for Virtual Brain Environments* [Arrouët 05, Renard 10]). La figure 4.13 illustre l'application « Use-the-force », où l'utilisateur soulève des objets en imaginant des mouvements de pieds. Un autre cas d'utilisation de ces interfaces est la prédiction de trajectoire. Wessberg *et al.* parviennent à déduire la trajectoire de la main d'un primate de son activité neuronale [Wessberg 00], et à faire reproduire par des robots le mouvement effectué, comme illustré en figure 4.14. Il est donc possible d'inférer la cible visée avant même que le geste ne soit effectué. Bien que ce concept semble prometteur, l'encombrement conséquent du dispositif et la difficulté de la calibration du système sont encore des obstacles à son utilisation.

FIGURE 4.13 – L'application « Use-the-force » utilise une interface cerveau-ordinateur pour soulever des objets virtuels. Image tirée de [Renard 10].

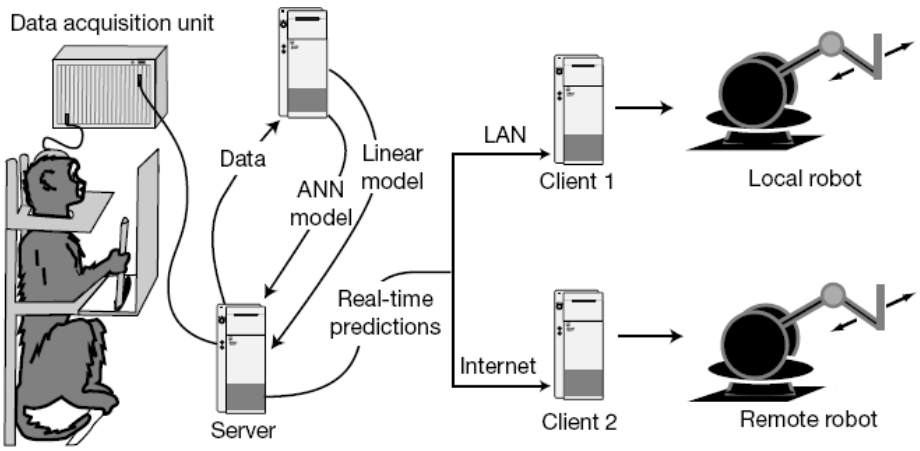


FIGURE 4.14 – Le singe effectue un geste. Un modèle permet de reproduire ce geste à l'aide uniquement de l'activité neuronale du singe. Image tirée de [Wessberg 00].

Prédire la cible d'un geste de sélection permettrait d'améliorer les performances d'un utilisateur en rendant obsolète (ou en facilitant) la phase de contrôle et la validation de la sélection. Cependant, le taux d'erreur encore élevé des algorithmes de prédiction ne permet pas de conclure précisément sur la position de la cible. À notre connaissance, l'algorithme KEP obtient les meilleures performances, pour un taux de réussite annoncé de 51%, pour un calcul de prédiction réalisé à 85% du mouvement. Nous estimons qu'un meilleur taux de réussite peut être obtenu en ne prenant pas en compte la phase balistique peu précise, mais uniquement la phase de contrôle. Nous proposons en section 4.5 une nouvelle approche pour une algorithme de prédiction pour un mouvement 2D. Cet algorithme sera détaillé dans le chapitre 7.

Un algorithme de prédiction ne garantit pas un taux de réussite parfait. Il est donc nécessaire de recourir à des techniques de sélection, afin de valider le choix de l'utilisateur. Dans les sections suivantes, nous nous intéressons dans un premier temps aux techniques basées sur le contrôle d'un avatar de la main de l'utilisateur dans la scène. Dans un second temps, nous verrons un état de l'art des techniques basées sur un rayon traversant la scène, et permettant de sélectionner à distance.

4.3 Avatar de la main

La recherche en réalité virtuelle a introduit de nombreuses techniques d'interaction afin de faciliter la sélection d'entités dans la scène virtuelle. Ces techniques peuvent être regroupées en deux catégories [Mine 95, Bowman 99b, Dang 07]. D'une part, certaines sont basées sur un rayon virtuel traversant la scène. L'utilisateur contrôle la trajectoire empruntée par ce rayon, et valide la sélection d'un objet atteint par le rayon, générale-

ment le plus proche. La deuxième catégorie des techniques de sélection utilise un avatar de la main de l'utilisateur. Celui-ci contrôle la position de l'avatar à l'aide d'un périphérique d'interaction, et valide la sélection d'un objet atteint (touché) par l'avatar. Dans cette section, nous verrons un état de l'art des techniques basées sur un avatar de la main. Nous nous intéresserons aux rayons dans la section 4.4.

L'avatar de la main est la métaphore la plus intuitive pour la sélection dans une scène virtuelle. La position d'un pointeur 3D est liée à la position du périphérique manipulé par l'utilisateur. Pour sélectionner, celui-ci doit atteindre précisément la cible avec le pointeur. Néanmoins, la taille de la scène dépasse généralement l'espace de travail fourni par le dispositif de réalité virtuelle. Certaines entités sont donc inaccessibles si l'utilisateur ne navigue pas dans la scène. Une autre solution est de modifier le *mapping* entre les positions du périphérique et du pointeur 3D. La technique Go-Go différencie deux types de mouvements en fonction de la distance entre le corps et la main de l'utilisateur [Poupyrev 96]. Lorsque la main est proche du corps, l'avatar est confondue avec elle, assurant un contrôle précis. Lorsque la main s'éloigne du corps, la distance de l'avatar n'est plus linéaire, mais quadratique en fonction de la distance de la main, permettant d'augmenter l'espace de travail, comme illustré en figure 4.15, au détriment de la précision. De manière analogue, la technique PRISM (*Precise and Rapid Interaction through Scaled Manipulation*) se base sur la vitesse de la main de l'utilisateur, et non sa position, pour contrôler la vitesse de l'avatar [Frees 05]. En dessous d'un certain seuil, la vitesse de l'avatar est inférieure à la vitesse de la main, permettant un contrôle précis. À l'inverse, au-delà d'un autre seuil, la vitesse de l'avatar dépasse celle de la main, afin d'atteindre des zones distantes.

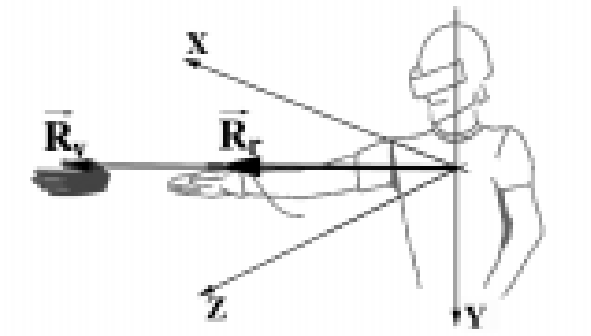


FIGURE 4.15 – Illustration de la technique Go-Go. Plus la main s'éloigne du corps, plus son avatar est placée à une grande distance. Image tirée de [Poupyrev 96].

Pour palier les problèmes de précision, les pointeurs utilisés sont généralement volumétriques et non ponctuels, à l'instar du Silk Cursor [Zhai 94], un parallélépipède semi-transparent. Dans une scène 2D, le Bubble Cursor est un disque dont le rayon évo-

lue dynamiquement pour n'atteindre qu'une seule cible [Grossman 05]. Cette technique a ensuite été étendue dans les environnements 3D, en utilisant une sphère [Vanacken 09] (voir la figure 4.16). Plusieurs études ont montrées que le Bubble Cursor est efficace dans les scènes peu denses, 2D et 3D [Vanacken 09, Blanch 11]. Cependant, dans les scènes denses, les différences entre le Bubble Cursor et un pointeur ponctuel semblent s'amenuiser. Des variantes du 3D Bubble Cursor ont été développées afin d'ajouter une information de profondeur [Rosa 10]. Néanmoins, bien que ces variantes permettent de mieux appréhender la position de la scène, aucune amélioration de performance en termes de temps de sélection ou de taux d'erreur n'a été observée.

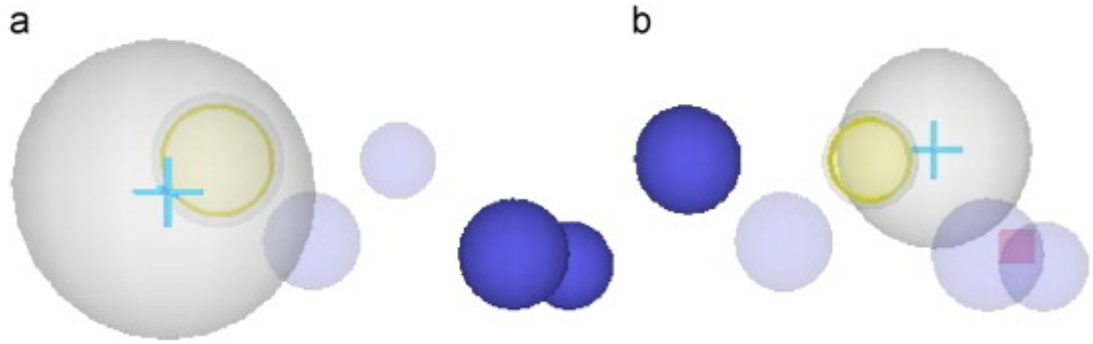


FIGURE 4.16 – Illustration du 3D Bubble Cursor. La taille de la sphère varie pour n'englober que la cible la plus proche. Image tirée de [Vanacken 09].

Dans le cas d'une scène de grande taille, il peut être intéressant d'utiliser un modèle World-in-Miniature [Stoakley 95] et de sélectionner directement dans ce modèle. Cependant, les environnements de ce type sont très denses, avec des objets de petite taille. Pour les scènes denses, la technique Ballon Selection exploite le mouvement des doigts de l'utilisateur pour contrôler un ballon virtuel [Benko 07], comme le montre la figure 4.17. La distance entre les deux mains indique la hauteur du ballon, tandis qu'un mouvement du pouce permet de modifier sa taille. Néanmoins, cette technique nécessite que toute la scène soit à portée de main, limitant son intérêt dans des environnements immersif, et requiert une interface tactile.

Nous avons vu au chapitre 3 qu'une mauvaise perception de la profondeur est l'un des obstacles à la sélection en environnement virtuel. Certaines études se sont penchées sur l'emploi de guides pour aider l'utilisateur à atteindre sa cible, en particulier en intégrant une composante haptique. Le premier guide proposé par cette approche est l'aimant. Lorsque l'utilisateur approche son pointeur suffisamment près d'un objet de la scène, une force est appliquée au dispositif haptique pour l'attirer en un point précis. Les aimants ont initialement été développés pour une application de dessin 3D, où une grille régulière d'aimants permettait de définir les sommets d'un maillage créé par l'utilisa-

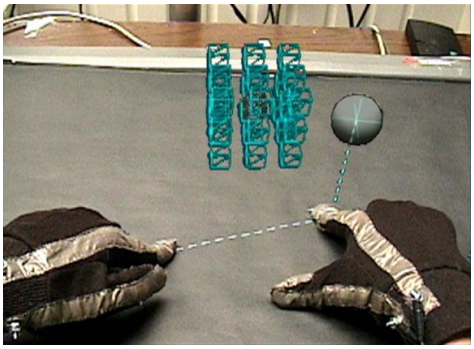


FIGURE 4.17 – Illustration de la technique Balloon Selection. La taille et la hauteur du ballon sont contrôlées par la position des doigts. Image tirée de [Benko 07].

teur [Yamada 02], comme le montre la figure 4.18. Cette méthode des aimants ponctuels, attirant l'utilisateur vers un point précis de l'espace, est principalement employée dans les applications de CAO. Cependant, dans ce type d'application, il peut également être intéressant de sélectionner non pas un point, mais une arête ou une face d'un objet, et de naviguer sur ces entités. Les aimants triples, illustrés en figure 4.19, définissent une zone d'attraction différenciée sur les faces, les arêtes et les sommets d'un objet [Picon 08]. Cette technique permet également de sélectionner un sommet en contraignant progressivement le pointeur de l'utilisateur, d'abord en un mouvement 2D sur une face, puis en un mouvement 1D sur l'arête, et enfin en l'immobilisant sur le sommet considéré. Les contraintes de mouvement procurées par les aimants permettent donc une sélection plus précise, aboutissant généralement à un gain de temps. Simard *et al.* généralise ce modèle d'aimant à l'aide d'un champ de potentiel [Simard 09]. Chaque entité topologique (sommets, arêtes et faces) génère un champ de potentiel. Le champ résultant de la somme de ces potentiels définit la force du retour haptique, comme l'illustre la figure 4.20.

Cet avantage peut cependant devenir un inconvénient, lorsque le pointeur est attiré par un aimant qui ne correspond pas à l'objet ciblé par l'utilisateur. Celui-ci devra donc forcer pour se libérer de son attraction, avec le risque d'être immédiatement attiré par un autre aimant proche, en particulier dans une scène dense. Cela se traduit généralement par une perte de confort d'utilisation, et une perte de temps en comparaison avec la sélection libre sans haptique. Un deuxième type de guide consiste à contraindre les mouvements de l'utilisateur sans l'attirer en permanence comme le ferait un aimant, mais en lui permettant de glisser sur une surface. La technique FOLLOW-ME utilise un ensemble de guides virtuels, uniquement visuels et non haptiques, afin de réaliser certaines tâches [Ouramdane 06]. Un cône, illustré en figure 4.21, contraint visuellement le pointeur de l'utilisateur pour le guider vers son extrémité, où se situe le point à sélectionner. L'ajout d'une composante haptique, en superposant par exemple un cône haptique au cône visuel, permet d'améliorer ces performances [Ullah 08].

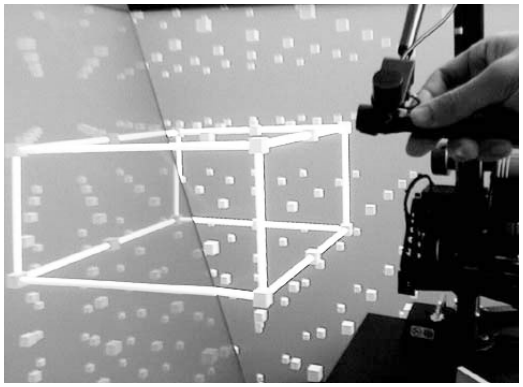


FIGURE 4.18 – Aimants haptiques. L'utilisateur peut créer un maillage en sélectionnant les sommets de la grille. Chaque point de la grille est un aimant. Image tirée de [Yamada 02].

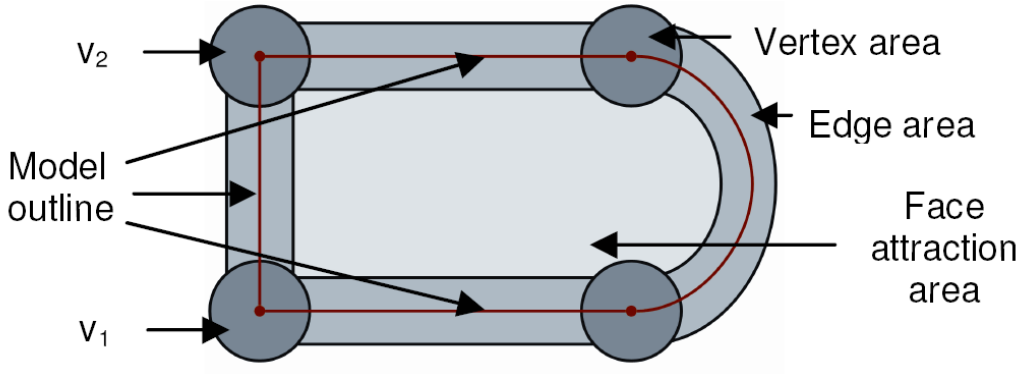
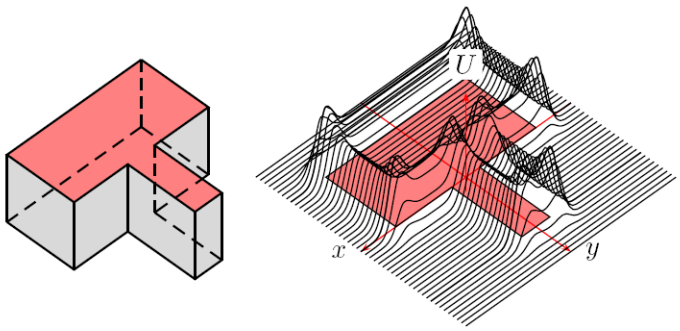


FIGURE 4.19 – Aimants triples. Les faces ont une aire plus grande, mais leur attraction est moins forte que les arêtes. De même, la force d'attraction des arêtes est surpassée par celle des sommets, ce qui permet de passer progressivement de la face à l'arête, puis de l'arête au sommet. Image tirée de [Picon 08].

L'utilisation d'un menu 3D est fréquente dans les applications de réalité virtuelle. La technique HardBorders exploite un menu polygonal dont les parois sont haptiquement activées [Essert-Villard 09], et dont les items sont placés dans les coins du polygone, comme illustré en figure 4.22. Le pointeur de l'utilisateur peut glisser sur les parois, et être aisément calé dans un coin pour sélectionner l'item correspondant. Cette assistance haptique permet une sélection plus confortable et plus rapide que les aimants, car l'utilisateur est simplement guidé vers les cibles, et non pas attiré brusquement sur elles.



(a) CAD scene (b) Potential field of the superior face

FIGURE 4.20 – Champ de potentiel associé à la face supérieure d'un objet de CAO. Les sommets génèrent une force plus élevée que les arêtes ou les faces. Image tirée de [Simard 09].

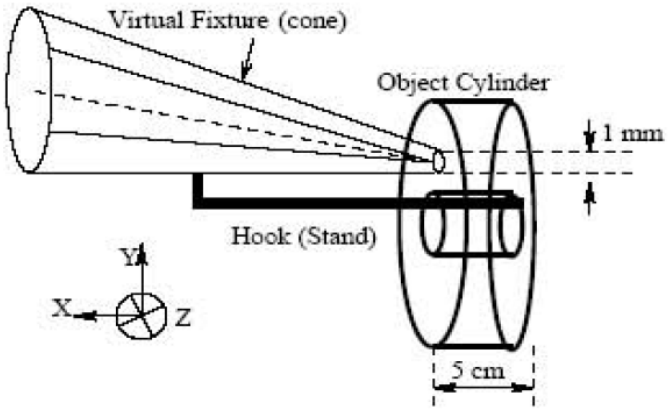


FIGURE 4.21 – Cône virtuel de la technique FOLLOW-ME. Le pointeur de l'utilisateur est contraint visuellement à l'intérieur du cône, lui permettant d'atteindre la cible à son extrémité. Image tirée de [Ouramdane 06].

Les techniques présentées dans cette section proposent une aide à la sélection par un pointeur 3D. Parmi ces techniques, celles guidant le mouvement du pointeur vers la cible à l'aide de contraintes haptiques ou visuelles, semblent apporter un plus grand confort d'utilisation et offrent de meilleures performances. Dans la section suivante, nous nous intéressons à la deuxième grande catégorie des techniques de sélection, celles basées sur un rayon.

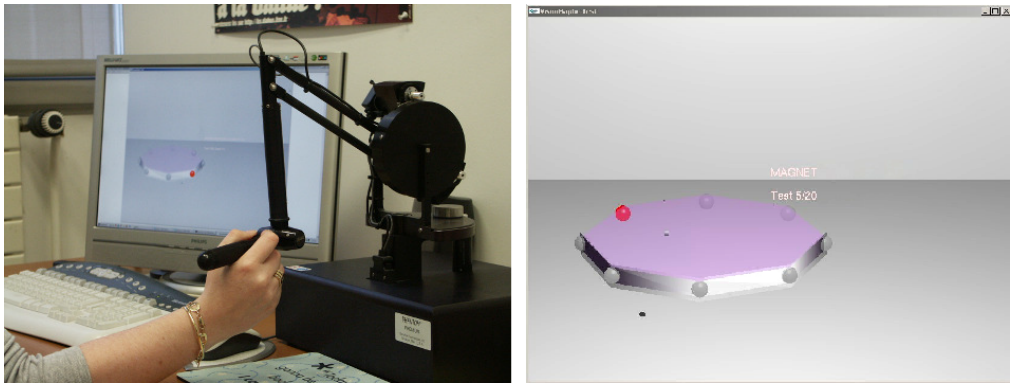


FIGURE 4.22 – Utilisation de HardBorders. **À gauche :** Le dispositif haptique manipule le pointeur à l'intérieur du menu. **À droite :** Le menu affiché est polygonal. Un item est placé à chacun de ses coins, et les parois ne peuvent pas être traversées. Images tirées de [Essert-Villard 09].

4.4 Rayons

Nous avons vu dans la section précédente que les techniques basées sur un avatar de la main permettent de sélectionner les objets de la scène proches de l'utilisateur. De manière générale, seule la zone de l'espace à portée de main est accessible, bien que certaines méthodes comme le Go-Go étendent cette zone. Les objets éloignées de l'utilisateur ne peuvent donc être sélectionnés qu'au prix d'une navigation dans la scène. Pour pallier cette limitation, des techniques basées sur la métaphore du rayon ont été développées. Intuitivement, un rayon virtuel part de la position de l'utilisateur et traverse la scène dans la direction souhaitée. Les cibles intersectées par ce rayon peuvent être sélectionnées. La comparaison de plusieurs techniques pour une même tâche a montré que la sélection d'un objet lointain par un rayon est plus rapide que par un avatar [Bowman 99a]. Cette section présente un état de l'art de cette catégorie de techniques.

En 1997, Pierce propose plusieurs méthodes de sélection ne tenant pas compte de la distance des objets, mais de leur projection dans le plan image [Pierce 97]. La technique Head Crusher, illustrée en figure 4.23, en est un exemple. La sélection est amorcée par la reconnaissance de deux doigts de l'utilisateur. Un objet dont la projection dans le plan image est située entre la projection de ces doigts est sélectionné. La position des doigts définit donc l'orientation et la taille d'un rayon, permettant de sélectionner le premier objet rencontré. Bien que cette technique soit intuitive, la sélection d'un objet peu ou non visible, et la distinction de deux objets proches dans le plan image, restent difficiles. Voodo Dolls étend les techniques proposées par [Pierce 97], et résout cette situation en affichant une miniature de l'objet sélectionné et des objets proches (voir la figure 4.24) près de la main non dominante de l'utilisateur, permettant la sélection et la manipulation de ces objets [Pierce 99].

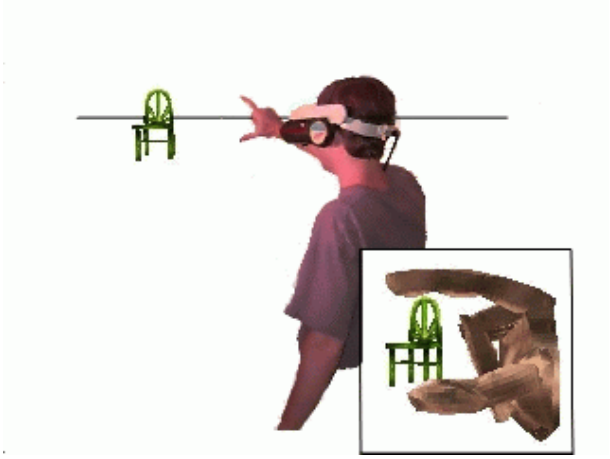


FIGURE 4.23 – Aperçu de Head Crusher. La technique reconnaît la position du pouce et de l'index dans le plan image, et sélectionne l'objet situé entre les deux doigts dans ce plan. La chaise est donc sélectionnée, même si sa position est éloignée de l'utilisateur dans l'espace virtuel. Image tirée de [Pierce 97].

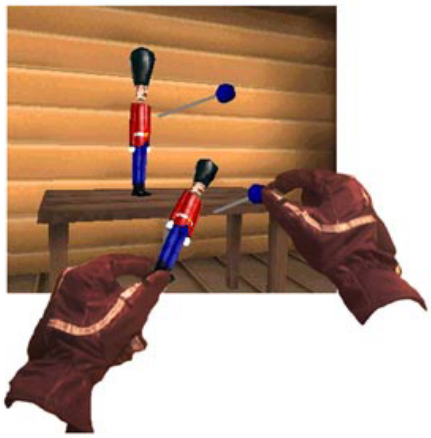


FIGURE 4.24 – Aperçu de Voodoo Dolls. Après la sélection d'un objet en exploitant le plan image, des miniatures de cet objet et des objets proches sont affichées près de la main non dominante de l'utilisateur. Image tirée de [Pierce 99].

Head Crusher et Voodoo Dolls demandent un alignement des yeux de l'utilisateur, de ses mains, et de l'objet à sélectionner. La technique Finger-gesture affiche un rayon virtuel au bout de l'index de l'utilisateur, dans la direction pointée par ce doigt [Song 00].

Cette solution permet un contrôle plus souple et une sélection plus rapide. Cependant, un rayon rectiligne nécessite une grande précision pour la sélection d'objets de petite taille. L'utilisation d'un cône réduit la précision requise, mais peut conduire à des ambiguïtés si plusieurs cibles se trouvent dans sa zone d'influence. Le Shadow Cone résout ces ambiguïtés en ne sélectionnant que l'objet conservé sans interruption dans la zone d'influence depuis le début de la tâche [Steed 04]. Autrement dit, en comparant la zone d'influence du Shadow Cone à un champ de lumière émis par une lampe, pour sélectionner un objet, il faut déplacer la lampe pour que cet objet soit le seul éclairé en permanence. La sélection est donc effectuée par raffinement progressif : l'ensemble des objets pouvant être sélectionnés est réduit jusqu'à n'obtenir qu'un seul élément. Bien que cette solution soit moins sensible aux tremblements et nécessite une moins grande précision, ses performances semblent diminuer lorsque la cible est proche de plusieurs distracteurs. De plus, il suffit que la cible sorte un court instant du champ d'action du cône pour qu'il ne soit plus possible de la sélectionner. La technique IntenSelect propose une alternative [De Haan 05]. À chaque instant, les objets situés dans la zone d'influence du cône augmentent leur score, tandis que celui des objets situés hors de la zone d'influence diminue. Visuellement, un rayon courbé relie la main de l'utilisateur à la cible de plus haut score, qui peut être sélectionnée, comme le montre la figure 4.25. IntenSelect semble permettre une sélection plus rapide qu'un cône dans le cas où les cibles sont statiques, mais reste moins performant qu'un rayon rectiligne [De Haan 05].

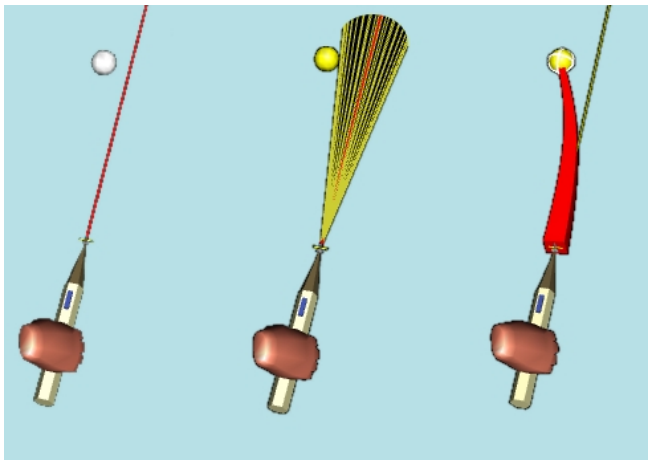


FIGURE 4.25 – Comparaison entre un rayon rectiligne, un cône de sélection, et IntenSelect. Le rayon d'IntenSelect est courbé pour atteindre la cible ayant le plus haut score, mais la zone d'influence est un cône (non visible). Image tirée de [De Haan 05].

Lever l'ambiguïté de la cible sélectionnée est un problème majeur. Plusieurs objets peuvent être situés dans la zone d'influence d'un cône, ou traversés par un rayon rectiligne. Dans ce deuxième cas, la pratique la plus courante est de sélectionner le premier

objet rencontré. Dans une scène très dense, il est donc très difficile, voire impossible, d'atteindre les objets les plus lointains. Le Depth Ray ajoute un marqueur de profondeur sur le rayon [Grossman 06]. La position de ce marqueur est liée à la position de la main et peut donc être déplacé en approchant ou en éloignant la main du corps. L'objet traversé par le rayon, et le plus près du marqueur de profondeur, peut être sélectionné, comme l'illustre la figure 4.26.

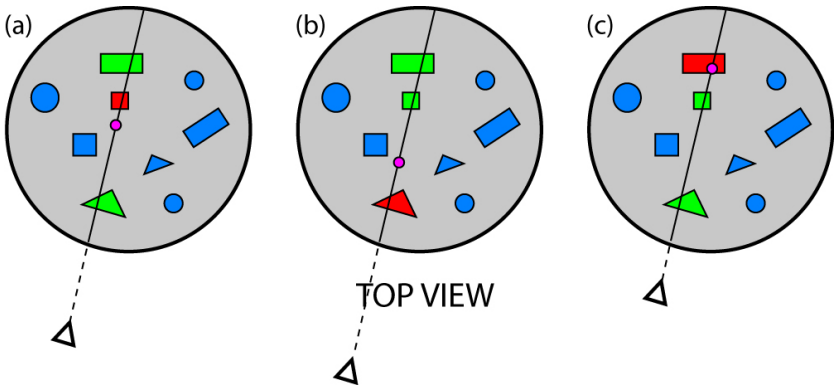


FIGURE 4.26 – Utilisation du Depth Ray. **(a)** L'objet traversé par le rayon, et le plus proche du marqueur de profondeur en rose, peut être sélectionné. **(b)** En reculant le dispositif d'interaction, un autre objet plus proche peut être sélectionné. **(c)** En avançant le dispositif d'interaction, un autre objet plus lointain peut être sélectionné. Image tirée de [Grossman 06].

Néanmoins, pour certaines scènes très denses, une autre approche semble nécessaire. Nous avons vu précédemment que le Shadow Cone opérait par raffinement progressif, de manière continue avec le mouvement du dispositif d'interaction. La technique SQUAD (*Sphere-casting refined by QUAD-menu*) propose un raffinement discret en filtrant progressivement les objets à l'aide d'un menu [Kopper 11]. Dans un premier temps, un rayon est lancé dans la scène virtuelle, projetant une sphère sur la première surface rencontrée. Les objets contenus dans la sphère sont alors répartis en quatre groupes dans un menu. L'utilisateur sélectionne le quadrant dans lequel est située la cible qui l'intéresse. Les objets de ce quadrant sont alors répartis dans un nouveau menu, et ainsi de suite jusqu'à la sélection d'un quadrant ne contenant qu'un seul élément. Cette procédure est illustrée en figure 4.27. SQUAD présente le défaut de perdre le contexte de la scène. L'utilisateur doit alors retrouver sa cible dans le menu, augmentant le temps requis pour la tâche, en particulier si plusieurs objets se ressemblent. La technique Expand étend SQUAD en affichant les objets présélectionnés dans une grille [Cashion 12], après une animation faisant la transition entre la position de ces objets dans la scène, et leur position dans la grille (voir figure 4.28). La conservation du contexte semble aboutir à une sélection plus rapide.

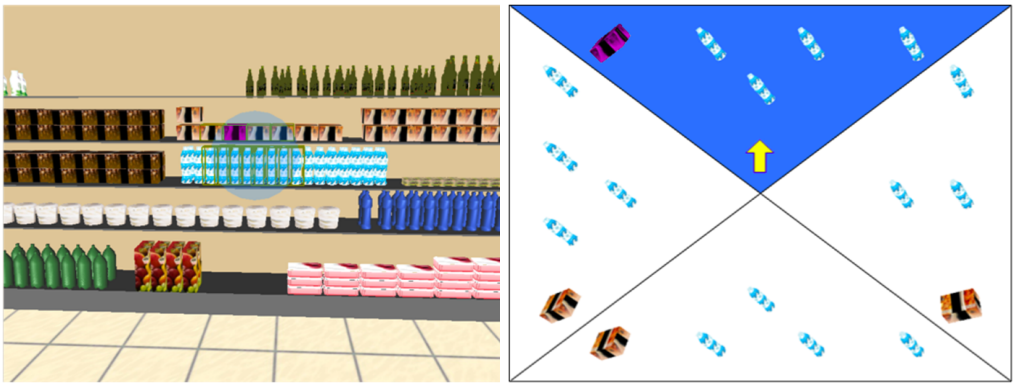


FIGURE 4.27 – Utilisation du SQUAD. **Gauche :** Une sphère est lancée dans la scène. **Droite :** Les objets dans la sphère sont répartis en quatre groupes dans un menu. La sélection est progressivement raffinée à chaque étape. Images tirées de [Kopper 11].

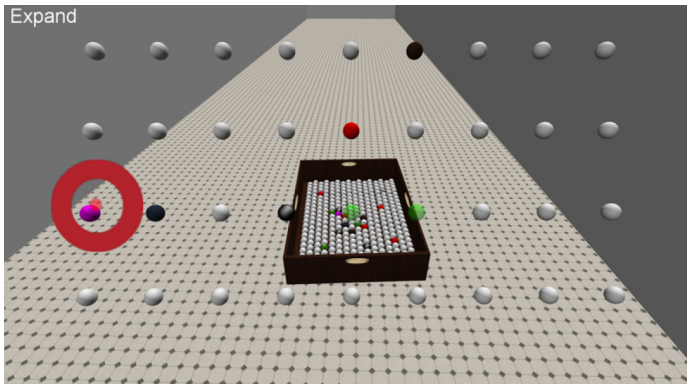


FIGURE 4.28 – Technique Expand. Des miniatures des objets pouvant être sélectionnées sont créées, et placées dans une grille après une animation. Image tirée de [Cashion 12].

La plupart des techniques basées sur un rayon ont pour origine le dispositif d'interaction dans la main de l'utilisateur. Or, l'ensemble des objets « visibles » depuis la main est différent de l'ensemble des objets visibles depuis les yeux de l'utilisateur (voir figure 4.29). La technique RCE (*Ray Casting from the Eye*) propose de lancer un rayon depuis les yeux de l'utilisateur, sa main étant utilisée pour contrôler la direction de ce rayon [Argelaguet 09]. Cette méthode semble aboutir à un gain de temps significatif et à moins de validations de sélection erronées, comparée à un rayon depuis la main.

Dans cette section et les précédentes, nous nous sommes attachés à présenter un état de l'art des différentes techniques aidant un utilisateur dans le processus de la sélection d'une cible. Plusieurs approches ont été abordées : aide à la localisation d'une cible,

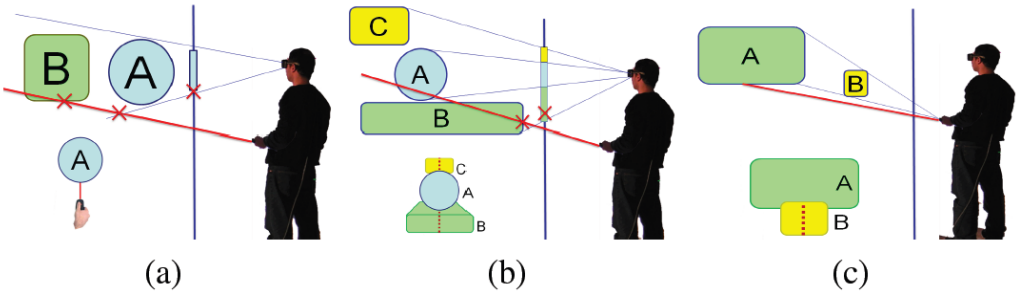


FIGURE 4.29 – Discordance entre les objets vus de la main, et ceux vus des yeux de l'utilisateur. **(a)** L'utilisateur pense sélectionner l'objet A, alors qu'il sélectionne l'objet B qu'il ne voit pas. **(b)** L'utilisateur voit l'objet C, mais ne peut pas le sélectionner car les objets A et B gênent le rayon partant de sa main. **(c)** L'objet A est visible depuis la main et les yeux, mais aucun point de son contour n'est visible simultanément depuis ces deux positions. Image tirée de [Argelaguet 09].

prédiction du mouvement, et assistance à la sélection. Dans la section suivante, nous présentons un aperçu de trois techniques que nous avons développées, une pour chacune de ses approches, en précisant les raisons qui nous ont menés à ce développement. Ces techniques seront ensuite détaillées dans la partie III.

4.5 Aides proposées

Dans le cadre de notre travail, nous nous sommes intéressés à l'amélioration des performances d'un utilisateur dans une tâche de sélection. Ce travail nous a conduit à développer trois nouvelles méthodes, basées sur trois étapes du processus de sélection. Dans cette section, nous décrivons ces méthodes et les motivations sous-jacentes qui ont mené à leur développement.

4.5.1 Localisation de cible

Comme nous l'avons vu précédemment, pour sélectionner une cible dans une scène virtuelle, il est nécessaire de la localiser au préalable. Dans la section 4.1, nous avons établi qu'une aide égocentrique semble préférable pour cette tâche, le changement de point de vue induit par les aides exocentriques comme les WIM entraînant une surcharge mentale et une réduction du sentiment d'immersion de l'utilisateur dans l'environnement. Parmi ces aides égocentriques, la technique la plus couramment employée est la flèche 3D. Généralement placée en face de l'utilisateur dans la scène virtuelle, cette flèche pointe, en fonction de l'application, vers le lieu ou l'objet d'intérêt le plus proche ou le plus pertinent. Il est à noter que plusieurs lieux ou objets peuvent être simultanément désignés en affichant plusieurs flèches. Cette pratique courante a cependant le défaut d'encombrer l'espace visuel, et nécessite d'insérer des indications pour identifier les entités pointées.

D'autre part, localiser un objet et naviguer vers lui ne sont que rarement l'objectif d'une application. La tâche assignée à l'utilisateur comporte souvent une sélection, puis une manipulation de cet objet. Ceci implique l'ajout d'une technique de sélection comme un rayon ou un avatar de la main. Nous estimons que l'affichage simultané de ces deux techniques encombre la scène et disperse la concentration de l'utilisateur. Cette conclusion nous a menés à notre ligne directrice : concevoir une aide visuelle fournissant une information directionnelle, et l'intégrer à une technique de sélection existante. La nouvelle technique résultante permettra ainsi de guider l'utilisateur jusqu'à sa cible, puis de la sélectionner. Cette aide visuelle devra être suffisamment souple pour être intégrée à différents outils que l'utilisateur pourrait manipuler.

Dans le chapitre 5, nous décrivons cette aide, le **Ring Concept**, basée sur l'affichage de plusieurs anneaux orientés dans la même direction. Nous présentons également le **Bubble Bee**, résultat de l'intégration du Ring Concept sur une sphère représentant un 3D Bubble Cursor, une des techniques de sélection obtenant les meilleures performances d'après plusieurs études.

4.5.2 Avatar de la main

Deux grandes familles de techniques ont été proposées afin de fournir à l'utilisateur une aide pour sélectionner une cible. Les méthodes basées sur un rayon sont particulièrement adaptées aux scènes peu denses, avec des cibles pouvant être situées à grande distance de l'utilisateur. Néanmoins, nous estimons que la précision requise pour sélectionner de telles cibles est très élevée et nécessite souvent un déplacement de l'utilisateur pour s'en rapprocher. À l'inverse, les techniques basées sur le contrôle d'un avatar de la main permettent une sélection rapide dans l'environnement proche. Cependant, une densité élevée de la scène entraîne une augmentation de la précision nécessaire, impliquant un taux d'erreur élevé et de l'inconfort pour l'utilisateur. Dans le cadre de notre travail, nous souhaitons mettre au point un avatar permettant un contrôle précis quelle que soit la densité de la scène. Le confort d'utilisation, bien qu'étant une notion subjective, doit également être pris en compte dans la conception de notre technique.

Dans le chapitre 6, nous décrivons cette technique, **Starfish** (*Selection of TARgets from Implicit Surfaces with Haptics*), basée sur la génération en temps réel d'une surface implicite fermée reliant un pointeur $3D$ aux cibles proches. Le pointeur, contraint dans les limites du volume formé par la surface, permet d'atteindre ces cibles sans exiger de gestes précis de l'utilisateur.

4.5.3 Prédiction de fin de mouvement

De nombreux algorithmes ont été développés pour estimer la cible d'un geste de sélection à partir des caractéristiques du mouvement. La plupart d'entre eux concerne la sélection dans une scène en deux dimensions, en utilisant un périphérique de type souris. Cependant, ces algorithmes considèrent que l'utilisateur effectue un geste de sélection

rectiligne. De fait, ils exploitent les caractéristiques du mouvement pour estimer a priori la distance totale que l'utilisateur va parcourir. La position dans la scène de la fin du geste est alors déterminée par sa direction moyenne. Nous pensons que pour sélectionner précisément une cible, l'utilisateur n'effectue rarement qu'un seul geste rectiligne. En particulier, si la distance à parcourir est grande, la phase balistique, sur laquelle s'appuient les algorithmes de prédiction existants, n'atteint généralement pas sa cible. Analyser une partie de la phase de contrôle semble alors nécessaire pour obtenir un meilleur taux de réussite. En effet, il semble que la phase d'accélération et la phase de décélération de l'impulsion initiale ne soient pas symétriques. Nous estimons que la distance prédite par KEP, l'algorithme de prédiction existant le plus performant, en modélisant l'intégralité du profil de vitesse par une courbe quadratique, peut être plus précise en ne modélisant que la phase de décélération, c'est-à-dire le début de la phase de contrôle. Nous souhaitons ainsi élaborer un nouvel algorithme de prédiction prenant en compte les corrections de vitesse et de trajectoire, et améliorant ainsi le taux de réussite des techniques existantes.

Dans le chapitre 7, nous détaillons cet algorithme, **SPEED** (*Speed Profile sEparation for Endpoint Divination*), basé sur une modélisation par une courbe quadratique de la phase de contrôle du profil de vitesse. Cette modélisation permet d'estimer la distance de la fin du mouvement. La position de la cible est obtenue en prenant en compte l'historique de la trajectoire parcourue.

4.6 Conclusion

La réalité virtuelle permet d'immerger un utilisateur dans une scène virtuelle. Dans de nombreuses applications, la tâche principale de l'utilisateur est généralement de manipuler certaines entités. Sélectionner ces entités est donc une étape obligatoire et importante. Ce domaine a déjà fait l'objet de nombreuses recherches qui ont permis de définir les problématiques inhérentes à cette tâche. Nous citerons en particulier le problème de la précision, d'autant plus nécessaire que la scène est densément peuplée, et que les objets présents sont petits. Des techniques ont été mises au point pour pallier ces difficultés, aidant l'utilisateur dans sa tâche par différentes approches. Nous avons abordé les motivations et le raisonnement qui ont mené à l'élaboration de trois nouvelles techniques aidant l'utilisateur dans les différentes étapes du processus de sélection.

La partie III suivante est consacrée à la description détaillée de ces apports. Le chapitre 5 présente le **Ring Concept**, une aide visuelle pour localiser des cibles dans la scène, et le **Bubble Bee**, une technique issue de cette aide. Le chapitre 6 détaille la technique de sélection **Starfish**, un pointeur 3D exploitant une surface implicite pour guider l'utilisateur. Le chapitre 7 décrit l'algorithme **SPEED**, qui analyse le mouvement de l'utilisateur pour en estimer la cible. Dans chacun de ces chapitres, nous présentons en particulier les outils théoriques utilisés et la mise en œuvre de la technique correspondante. Nous mettons également en place un protocole expérimental, quantifiant l'aide apportée par notre méthode, comparée aux techniques existantes.

Troisième partie

Aides à la sélection

Introduction

Dans le cadre de notre travail, nous nous intéressons principalement aux problématiques liées à la sélection de cibles en environnement de réalité virtuelle. Dans ce contexte, un utilisateur est soumis à plusieurs obstacles que nous avons détaillés dans la deuxième partie de ce document (voir Chapitre 3). En particulier, une mauvaise perception de la profondeur fausse la compréhension de la scène. Par ailleurs, une scène dense entraîne des phénomènes d'occlusion entre les objets, perturbant la localisation de la cible. De plus, une telle scène requiert une grande précision de la part de l'utilisateur pour atteindre et donc sélectionner la cible qu'il souhaite. Cette difficulté est accrue par les tremblements naturels, liés à la fatigue, que produit la main actionnant le périphérique d'interaction, en particulier après de longues séances de manipulation. Notre objectif est de proposer, de mettre en place et d'évaluer des techniques de sélection pour pallier ces obstacles.

Dans cette partie, nous nous intéressons à trois approches d'aides à la sélection, que nous détaillons dans les trois prochains chapitres. Le Chapitre 5 présente le **Ring Concept**, une méthode apportant un indice visuel pour indiquer dans quelle direction se trouve une cible. La technique **Bubble Bee** est issue de l'intégration du Ring Concept à la technique de sélection 3D Bubble Cursor. Dans le Chapitre 6, nous présentons la technique **Starfish**, simplifiant la sélection en contraignant visuellement le pointeur de l'utilisateur vers la cible, à l'aide de guides virtuels. Enfin, dans le Chapitre 7, nous étudions et modélisons le profil de vitesse d'un geste de sélection dans une scène 2D, en séparant la phase balistique de la phase de contrôle. L'algorithme que nous dérivons de ce modèle, **SPEED**, analyse le mouvement de l'utilisateur et estime la position de fin du geste, et donc la cible visée.

Nous avons également mis en place trois expérimentations qui nous permettent de quantifier l'aide apportée par les trois techniques que nous proposons, comparées aux techniques les plus couramment utilisées et les plus performantes dans leur domaine respectif.

Chapitre 5

Ring Concept et Bubble Bee

One Ring to rule them all,
One Ring to find them,
One Ring to bring them all
and in the darkness bind them

John Ronald Reuel Tolkien,
The Lord of the Rings

Sommaire

5.1 Principe du Ring Concept 76
5.2 Mise en œuvre du Bubble Bee 78
5.3 Évaluation du Ring Concept 80
5.3.1 Matériel et participants 81
5.3.2 Procédure 81
5.3.3 Conception 83
5.3.4 Difficulté de la tâche 84
5.4 Résultats 85
5.4.1 Temps de décision 85
5.4.2 Niveau d'erreur 86
5.5 Discussion 87
5.6 Conclusion 88

L'un des intérêts de la réalité virtuelle est de simuler un monde 3D, qu'il soit inspiré de notre monde réel ou qu'il soit imaginaire. Ces mondes peuvent être réduits à un petit espace de travail ou au contraire être très vastes, nécessitant alors une phase d'exploration et de navigation pour effectuer les tâches demandées par l'application. Des indications peuvent alors être données à l'utilisateur pour localiser la position d'un point d'intérêt,

par exemple un objet avec lequel il peut interagir, ou la sortie d'un labyrinthe. Dans le monde réel, les panneaux arborant une flèche sont omniprésents pour nous guider, comme sur les routes pour nous conduire vers une ville, ou dans les magasins pour indiquer la position d'un rayon.

Dans les mondes virtuels, d'autres outils ont été mis en place, comme les radars pour indiquer simultanément la position de différents objets dans la scène par rapport à l'utilisateur. Toutefois, pour pointer une direction dans l'espace, la flèche, 2D ou 3D selon les applications, reste l'une des méthodes les plus simples et les plus efficaces. Cependant, dans les conditions habituelles d'utilisation d'une application de réalité virtuelle, plusieurs outils virtuels peuvent être affichés simultanément. Il est en effet courant d'afficher, en plus de la scène virtuelle elle-même, un outil pour interagir avec les objets de la scène comme un rayon, des menus, diverses indications comme le temps restant pour accomplir une tâche, ... Cette accumulation d'informations disperse l'attention de l'utilisateur, contribue à densifier l'espace visuel et augmente le nombre d'occlusions. Or, comme nous l'avons vu dans le Chapitre 3, les occlusions dans une scène sont un obstacle à la localisation des différents objets. Nous souhaitons donc développer une technique permettant de pointer une direction sans occuper d'espace visuel supplémentaire, au contraire d'une flèche 3D.

Dans le cadre de notre travail, nous nous intéressons à la tâche de sélection. Avant de sélectionner un objet, il est nécessaire de le visualiser, ou de localiser sa position, afin de planifier son geste de sélection. Nous sommes donc typiquement dans le cas ou une technique de sélection et une technique de pointage de direction sont nécessaires simultanément. Parmi les techniques de sélection existantes, le 3D Bubble Cursor [Grossman 05, Vanacken 09] est l'une des plus efficaces. Nous rappelons qu'il s'agit d'une sphère dont le rayon varie dynamiquement pour atteindre l'objet le plus proche, et uniquement celui-ci, permettant de le sélectionner rapidement. Cette technique est intrinsèquement source de nombreuses occlusions. Si le centre de la sphère est éloigné de tout objet, alors son rayon est élevé. Dans ces conditions, nous souhaitons éviter d'ajouter une flèche 3D pour indiquer la direction d'une cible particulière, par exemple celle que l'application demande à sélectionner pour accomplir une certaine tâche. En effet, l'affichage de ces deux techniques pourrait gêner la perception de l'utilisateur et disperser sa concentration, détériorant ainsi ses performances. Notre objectif est de concevoir une nouvelle aide visuelle à la localisation, fournissant une indication de direction à l'instar de la flèche 3D, pouvant être directement et visuellement intégrée aux outils manipulés par l'utilisateur comme un 3D Bubble Cursor. En d'autres termes, nous souhaitons augmenter la technique de sélection avec une information directionnelle.

Dans ce chapitre, nous présentons tout d'abord le principe que nous appelons le **Ring Concept**. Cette méthode se base sur l'affichage de plusieurs cercles, ou *anneaux*, afin de pointer une direction. Après avoir expliqué le principe mathématique sous-jacent, nous intégrons le Ring Concept à la technique de sélection la plus couramment employée et

la plus performante, le 3D Bubble Cursor. La technique résultante, appelée **Bubble Bee**, est ensuite évaluée par un protocole expérimental que nous détaillons et dont nous présentons les résultats.

5.1 Principe du Ring Concept

L'idée la plus simple afin de représenter un axe est d'afficher un segment. La pointe d'une flèche peut être ajoutée à l'extrémité de ce segment pour fixer une direction donnée par cet axe. Cependant, un utilisateur observant ce segment n'aura qu'une perception biaisée de la direction indiquée, car il n'en voit à tout instant qu'une projection sur un écran. En effet, la longueur visible du segment est différente selon le point de vue de l'utilisateur. Comme l'illustre la figure 5.1 (en **Haut**), l'utilisateur ne perçoit la longueur réelle du segment que lorsque son regard lui est orthogonal. Sa longueur visible diminue lorsque l'utilisateur modifie son point de vue, jusqu'à n'être plus qu'un point quand le regard et le segment sont parallèles. À moins de se déplacer afin d'observer le segment sous différents points de vue, il n'est pas possible de savoir avec certitude si le segment est orthogonal au regard, ou s'il s'agit d'un plus grand segment vu sous une autre orientation. Cette ambiguïté peut être levée à l'aide d'une longueur de référence. Dans la figure 5.1 (en **Bas**), la flèche bleue effectue une rotation dans un plan orthogonal à la flèche rouge qui elle, reste immobile. Sans cette flèche de référence, on ne peut pas déterminer la longueur et l'orientation de la flèche bleue.

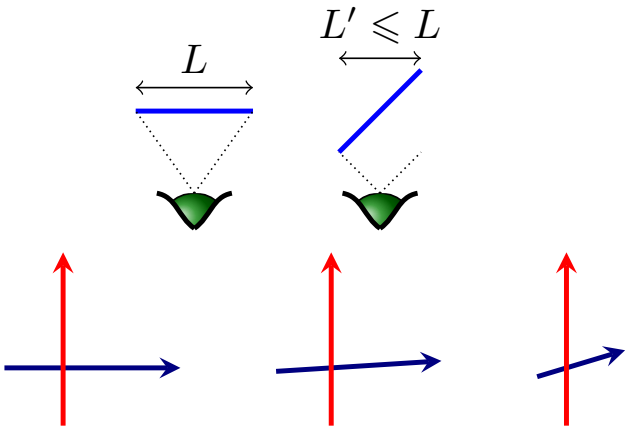


FIGURE 5.1 – Perception d'un segment. **Haut :** Quand le segment est orthogonal au regard, l'utilisateur perçoit la longueur réelle L du segment. Après rotation, le segment paraît plus petit. La taille L' perçue dépend de l'angle de rotation θ : $L' = L\cos(\theta)$. **Bas :** Sans la longueur de référence constante (en rouge), il n'est pas possible de déterminer la taille et l'orientation de la flèche bleue. La présence de la longueur de référence indique la longueur réelle de la flèche bleue, nous permettant d'inférer son orientation.

Nous émettons l'hypothèse que l'information directionnelle fournie par une flèche 3D provient des extrémités circulaires du tube et du cône de la flèche. En effet, un cercle (ou anneau) est perçu comme une ellipse, en fonction de son orientation et du point de vue de l'utilisateur, comme l'illustre la figure 5.2. La longueur du demi-grand axe est égale à la longueur du rayon de l'anneau, et sert donc de longueur de référence. La comparaison du demi-petit axe et du demi-grand axe de l'ellipse permet de déterminer deux orientations possibles. Mathématiquement, il existe une relation entre ces deux longueurs et l'angle de rotation de l'anneau. Soit $\mathcal{C}$ un cercle dans un plan $\mathcal{P}$, de rayon a. Appliquons une rotation à $\mathcal{C}$, d'angle θ et d'axe un diamètre du cercle, et appelons ce nouveau cercle $\mathcal{C}'$. La projection de $\mathcal{C}'$ sur $\mathcal{P}$ est une ellipse, dont la longueur du demi-grand axe est a, et la longueur du demi-petit axe est $b = a\cos(\theta)$. La connaissance des deux longueurs a et b permet d'aboutir à deux solutions pour θ. Cette ambiguïté est cohérente avec le fait qu'un anneau seul peut être interprété comme un cercle orienté dans deux directions différente. Ce phénomène est illustré en figure 5.3.

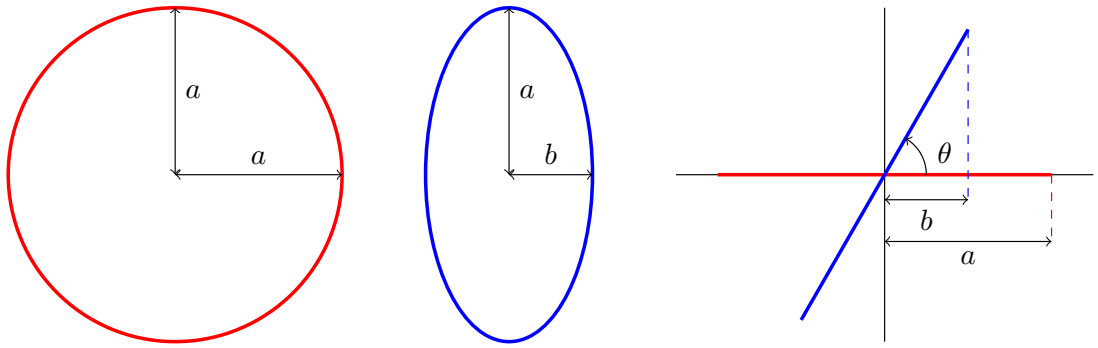


FIGURE 5.2 – Longueur du demi-petit axe de l'ellipse perçue après la rotation d'un cercle. **Gauche :** Le cercle est orthogonal au regard. L'utilisateur perçoit bien un cercle. **Centre :** Après une rotation du cercle, l'utilisateur perçoit une ellipse. Le demi-grand axe de cette ellipse a la même longueur que le rayon du cercle. **Droite :** Vue de dessus du cercle et de l'ellipse. Deux angles de rotation peuvent être inférés des longueurs du demi-grand axe et du demi-petit axe de l'ellipse : $\theta = \pm\arccos\left(\frac{b}{a}\right)$. Une ellipse seule peut être le résultat de deux rotations : une d'angle θ et une autre d'angle $\pi - \theta$.

Si un unique anneau ne permet pas de choisir parmi ces deux orientations potentielles, des occlusions peuvent aider à lever l'ambiguïté. Sur les deux cylindres de la figure 5.3, l'information directionnelle est donnée par les deux anneaux à leurs extrémités. Seule l'occlusion d'un des anneaux (en pointillés) permet de distinguer les deux orientations possibles. Nous pensons que l'indication de direction peut être renforcée en affichant plusieurs anneaux, orientés dans la même direction, et alignés dans cette direction, et que cette redondance peut raccourcir le temps requis pour interpréter correctement l'indication fournie.

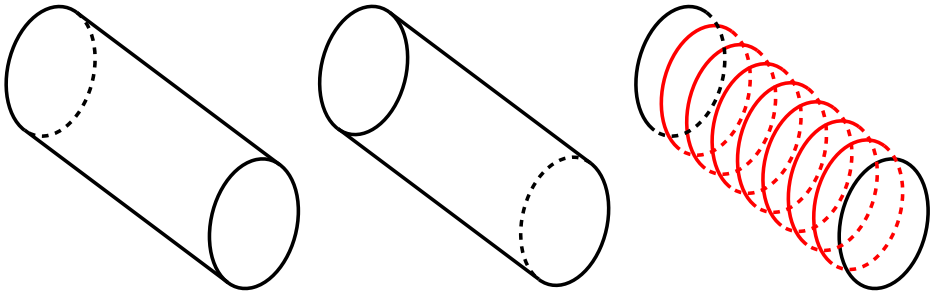


FIGURE 5.3 – Redondance de l'information directionnelle. **Gauche et centre :** Deux cylindres presque identiques. Leur unique différence est l'occlusion qui n'affecte pas la même extrémité. Cette différence suffit pour distinguer les deux directions qui peuvent être interprétées par un anneau unique. **Droite :** L'information directionnelle est propagée tout le long du « cylindre » par les anneaux.

Ces anneaux définissent un axe. Notre objectif étant d'indiquer une direction, il est nécessaire de distinguer les deux sens de l'axe. Nous avons choisi d'utiliser un code couleur simple, le sens rouge-vert donnant la « bonne » direction, comme illustrée en figure 5.4.

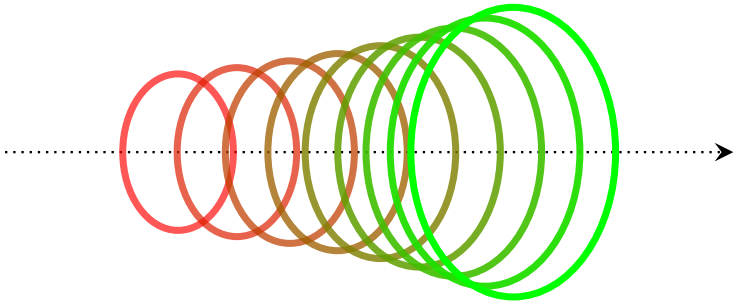


FIGURE 5.4 – Code couleur du Ring Concept. Le degradé du rouge vers le vert précise le sens de l'axe défini par les anneaux.

5.2 Mise en œuvre du Bubble Bee

Nous avons pensé le Ring Concept afin de l'intégrer visuellement à différents outils manipulés par l'utilisateur, lui procurant une information directionnelle. Nous disons que l'outil en question est *augmenté* par le Ring Concept. Dans le contexte qui nous intéresse, nous souhaitons naturellement valider notre approche en augmentant une technique de sélection. Dans une scène 3D, l'une des techniques les plus performantes et les plus

couramment employées est le 3D Bubble Cursor. Rappelons qu'il s'agit d'un pointeur sphérique dont le rayon s'adapte dynamiquement pour atteindre l'objet le plus proche, et uniquement celui-ci. Un rayon maximal est bien sûr défini afin que le pointeur conserve une taille raisonnable, même si aucun objet n'est à proximité. Il serait en effet sans intérêt de laisser le pointeur grossir indéfiniment, obstruant ainsi tout l'espace visuel. En utilisant cette technique, il peut être intéressant pour l'utilisateur de disposer d'une information pour localiser la cible la plus proche si le pointeur est éloigné de toute cible, ou pour localiser la cible la plus pertinente (selon un certain critère défini par l'application) si le 3D Bubble Cursor est déjà en contact avec une autre cible.

Nous souhaitons augmenter le 3D Bubble Cursor par le Ring Concept. La forme de base étant une sphère, une intégration simple peut être obtenue en ajoutant des anneaux sur la surface de la sphère, indiquant la direction d'une cible. Ces anneaux étant alignés et centrés sur le même axe, ils n'ont pas tous le même rayon. Afin de rendre le code couleur plus visible, nous avons choisi un dégradé continu du rouge au vert sur toute la surface de la sphère, comme illustré sur la figure 5.5. Nous appelons le résultat de cette intégration le **Bubble Bee**, ses rayures évoquant une abeille.

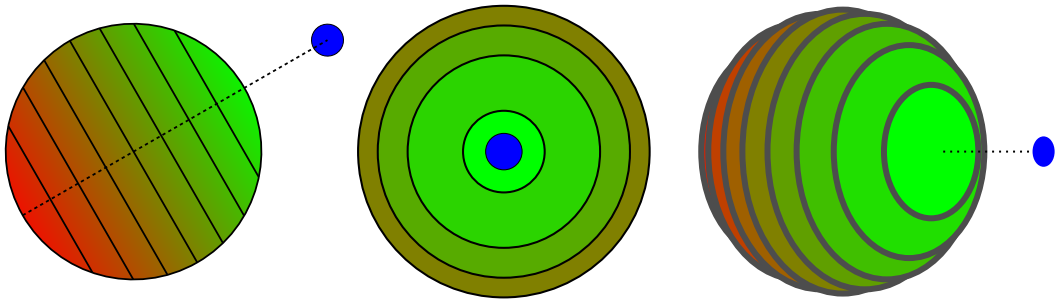


FIGURE 5.5 – Plusieurs vues du Bubble Bee. La cible (en bleu) est pointé par les anneaux.

Note : Dans la section suivante détaillant le protocole expérimental que nous avons mis en place pour évaluer le Ring Concept, toutes les cibles sont à même distance du Bubble Bee. Les anneaux sont alors tous dirigés vers une cible choisie aléatoirement. Ceci n'est pas une suggestion de comportement. L'augmentation par le Ring Concept fournit une information directionnelle, et la direction indiquée doit être choisie selon l'application.

5.3 Évaluation du Ring Concept

À travers le Ring Concept, nous souhaitons proposer une aide visuelle fournissant une indication directionnelle, dont un avantage serait d'être directement intégrée dans un outil manipulé par l'utilisateur, afin de ne pas nécessiter un outil supplémentaire comme la flèche 3D qui encombre l'espace visuel et disperse l'attention de l'utilisateur. Nous décrivons dans cette section le protocole expérimental que nous avons mis en place afin de quantifier l'aide apportée par le Ring Concept pour désigner un objet de la scène. Nous nous focalisons sur le temps nécessaire pour interpréter l'information directionnelle, et sur le taux d'erreur de cette interprétation. Nous choisissons d'utiliser le Bubble Bee comme support du Ring Concept, car le 3D Bubble Cursor est une technique bien connue. De plus, sa forme simple et isotrope permet d'éviter les influences sur l'information directionnelle que des protubérances pourraient provoquer. Nous estimons que la sphère est donc la forme idéale pour notre évaluation. Notre expérience portant sur la désignation d'objet, nous souhaitons comparer le Bubble Bee avec la technique la plus efficace et la plus utilisée pour cette tâche, la flèche 3D.

Une scène, représentée en figure 5.6, est composée d'un ensemble de cibles sphériques disposées régulièrement en une demi-sphère, devant l'utilisateur. La technique utilisée, le Bubble Bee ou la flèche 3D, est placée au centre de la demi-sphère, et pointe vers l'une des cibles. Nous avons choisi cette disposition afin que toutes les cibles soient à la même distance de la technique, et qu'aucune d'entre elles ne soient cachée par une autre. Nous souhaitons évaluer les performances de chaque technique dans différentes configurations de scène. Les deux paramètres que nous faisons varier pour créer nos scènes sont le rayon de la demi-sphère de cibles, et le nombre de cibles. Chacun de ses paramètres peut prendre trois valeurs, formant ainsi neuf scènes différentes.

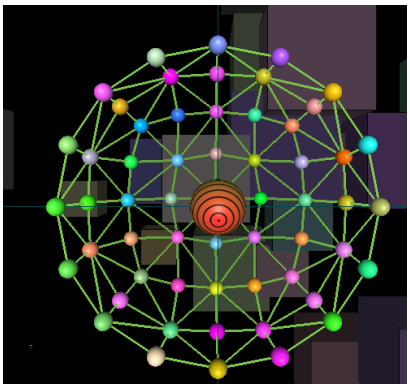


FIGURE 5.6 – Capture d'écran de la scène lors de l'évaluation du Bubble Bee. Des cibles sont disposées uniformément sur une demi-sphère. La technique pointe vers l'une d'elles. La tâche de l'utilisateur est de déterminer la cible désignée.

Lorsqu'une scène apparaît, l'utilisateur doit désigner la cible pointée par la technique. Bien que notre objectif ne soit pas d'évaluer une technique de sélection, une telle technique est nécessaire pour que l'utilisateur puisse faire son choix. Afin d'éviter des influences non souhaitables entre la technique de pointage évaluée et la technique de sélection, nous avons choisi de ne pas utiliser le Bubble Bee, mais un rayon virtuel, dont le maniement est bien connu des utilisateurs, et suffisamment précis dans les conditions de l'expérimentation.

5.3.1 Matériel et participants

Durant cette expérience, nous avons utilisé un mur immersif (3m $\times$ 2.25m) avec un affichage stéréoscopique actif, ainsi qu'une Nintendo Wiimote comme périphérique d'interaction. Douze (12) volontaires ont participé à cette expérience, 3 femmes et 9 hommes, entre 25 et 48 ans (pour une moyenne de 30 ans). Tous ces participants avaient déjà effectué au préalable plusieurs expériences dans des environnements immersifs.

5.3.2 Procédure

Il est demandé aux participants d'effectuer une tâche de sélection, divisée en deux séries, chacune correspondant à une technique, le Bubble Bee ou la flèche 3D. Pour chacune des 9 combinaisons (distance des cibles, densité de la scène), 9 sélections sont requises, pour un total de 81 scènes par série. Pour chaque série, toutes les scènes sont proposées dans un ordre aléatoire. Il est à noter que deux pauses sont autorisées par série, toutes les 27 scènes, ceci afin de garder l'utilisateur concentré pendant toute la durée de la tâche.

Les cibles sont de petites sphères, colorées aléatoirement. Par ce moyen, nous évitons à l'utilisateur de « perdre » du regard la cible qu'il souhaite désigner. Pour une scène donnée, la tâche se déroule de la manière suivante. Les cibles placées en demi-sphère apparaissent devant l'utilisateur. La technique de localisation, Bubble Bee ou flèche 3D, est placée au centre de la demi-sphère, et désigne l'une des cibles. Lorsque l'utilisateur pense avoir correctement interprété l'information donnée par la technique de localisation, et donc pense savoir quelle cible est pointée, il lui est demandé d'appuyer sur l'un des boutons de la Nintendo Wiimote. À ce moment, la technique de localisation disparaît, tandis qu'un rayon virtuel contrôlé par le périphérique d'interaction apparaît. Dans la suite de ce chapitre, nous disons que l'utilisateur est passé en phase de sélection. Aucune indication sur la cible précédemment désignée n'est alors disponible. L'utilisateur doit sélectionner la cible qu'il a choisie avec ce rayon, et une nouvelle tâche débute, avec l'apparition d'une nouvelle scène. Afin de ne pas disperser son attention, l'utilisateur n'est pas informé de la réussite ou non de son choix, *i.e.* s'il a bien sélectionné la cible désignée par la technique de localisation.

Dans cette expérience, nous ne nous intéressons pas aux performances de la technique de sélection. Nous nous focalisons sur le **temps de décision**, et le **niveau d'erreur**.

Nous définissons le temps de décision comme la durée entre le début d'une tâche, lorsqu'une scène apparaît, et le passage en phase de sélection par l'appui d'un bouton de la Nintendo Wiimote. Le temps de décision ne prend donc pas en compte le temps pour sélectionner la cible avec le rayon virtuel, mais seulement le temps pour l'utilisateur d'interpréter l'information fournie par la technique.

Le niveau d'erreur, que nous notons NE, est une quantité qui représente grossièrement le nombre de cibles séparant celle désignée par la technique de localisation, et celle sélectionnée par l'utilisateur. Cette quantité est calculée de la manière suivante. Pour chaque scène, nous calculons l'écart angulaire minimal $\delta_{\min}$ entre deux cibles de la demi-sphère :

$$\delta_{\min} = \min_{i \neq j} \widehat{C_i O C_j}, \tag{5.1}$$

où O est le centre de la demi-sphère où est placée la technique de localisation, et les $(C_i)_i$ sont les cibles. La répartition régulière des cibles implique que l'écart angulaire entre deux cibles adjacentes, *i.e.* sur une même corde horizontale ou verticale par rapport à l'utilisateur, vaut $\delta_{\min}$. Nous calculons ensuite l'écart angulaire δ_θ entre la cible T désignée par la technique, et la cible U sélectionné par l'utilisateur. Le niveau d'erreur est enfin défini comme le ratio entre δ_θ et $\delta_{\min}$:

$$NE = \frac{\delta_\theta}{\delta_{\min}} = \frac{\widehat{TOU}}{\min_{i \neq j} \widehat{C_i O C_j}}. \tag{5.2}$$

Le niveau d'erreur vaut 0 si l'utilisateur a bien sélectionné la bonne cible, 1 s'il a sélectionné une cible adjacente, et ainsi de suite. La figure 5.7, obtenue en post-traitement, illustre cette quantité. Chaque sphère est colorée selon son niveau d'erreur par rapport à la cible à atteindre (en blanc sur la figure).

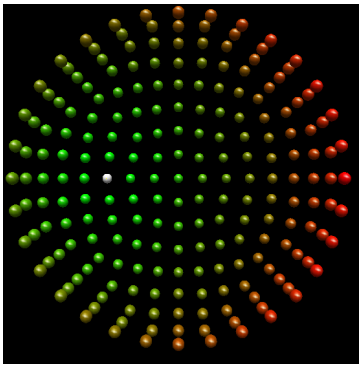


FIGURE 5.7 – Illustration en post-traitement du niveau d'erreur. La sphère blanche représente la cible T désignée par la technique. Les sphères vertes sont les cibles adjacentes à T ($NE = 1$). Les sphères rouges sont les plus éloignées de T ($NE = 11$ sur cette figure).

5.3.3 Conception

Les variables indépendantes de cette expérience sont :
- la technique de localisation **T** : Bubble Bee (**BB**) et la flèche 3D (**F3D**),
- le rayon de la demi-sphère de cibles **R** : petit (**PR**), moyen (**MR**) et grand (**GR**), et
- la densité de la scène **D** : faible (**DF**), intermédiaire (**DI**) et élevée (**DE**).

Notre objectif est d'évaluer la désignation de cibles à différentes distances et densités, en nous focalisant sur le temps de décision et le niveau d'erreur. Nous émettons les hypothèses suivantes sur les résultats :

H1 Nous prévoyons que le temps de décision augmente avec la densité de la scène. Il semble naturel de penser que lorsque le nombre de cibles augmente, l'utilisateur a besoin d'un délai plus important pour distinguer celle désignée par la technique.

H2 Nous prévoyons également que le temps de décision augmente avec la distance des cibles, c'est-à-dire le rayon de la demi-sphère.

H3 Nous pensons que le Bubble Bee est aussi intuitif que la flèche 3D, *i.e.* le temps de décision est similaire pour les deux techniques, quelle que soit la scène considérée.

H4 De même, nous estimons que le niveau d'erreur entre les deux techniques est similaire quelle que soit la scène considérée, *i.e.* pour chaque combinaison **R**/**D**, il n'y a pas de différence significative entre les techniques pour le niveau d'erreur.

Afin d'éviter les effets d'apprentissage, la moitié des utilisateurs ont débuté l'expérience avec le Bubble Bee, et l'autre moitié avec la flèche 3D. Pour une technique donnée, l'ordre des 81 scènes a été déterminé aléatoirement. Chaque participant a effectué l'expérience en une session d'approximativement 20 minutes, en commençant par une série d'entraînement où chaque technique, chaque rayon de demi-sphère et chaque densité de la scène sont présentés.

Plusieurs paramètres de cette expérience ont été fixés empiriquement afin que la tâche soit la plus confortable possible. Les distances des cibles et les densités de scènes ont été choisies afin de satisfaire un compromis entre présenter à l'utilisateur des situations réalistes, et évaluer les techniques sous plusieurs niveaux de difficultés. La technique de localisation est placée à $2,20$ m devant l'utilisateur, à $0,5$ m en dessous des yeux, afin que tous les participants aient le même point de vue de la scène. La demi-sphère de cibles, centrée sur la technique, a pour rayon $0,2$ m (**PR**), $0,45$ m (**MR**) et $0,70$ m (**GR**). La densité de la scène est représentée par le nombre de cibles affichées : 57 (**DF**), 105 (**DI**) et 209 (**DE**). Les cibles ont un rayon de 2 cm. Le Bubble Bee est une sphère de 6 cm de rayon. La flèche 3D est composée d'un cylindre rouge de 6 cm de longueur, et d'un cône vert de 2 cm de hauteur. Ces couleurs ont été choisies afin de respecter le code couleur du Bubble Bee, comme décrit dans la section précédente.

5.3.4 Difficulté de la tâche

L'indice de difficulté I_d défini dans la loi de Fitts (voir l'équation (3.2)) donne une estimation sur la difficulté d'une tâche de pointage, en fonction de la taille et de la distance d'une cible. Dans le cadre de cette évaluation, nous avons calculé cet indice pour prévoir les résultats et émettre nos hypothèses. Les valeurs numériques indiquées précédemment nous donnent des indices de difficulté de 3.46, 4.55 et 5.17 respectivement pour les trois rayons de demi-sphère. Ces valeurs sont récapitulées dans la table 5.1. Ces indices sont calculés en prenant une taille de cible constante. La densité n'intervient donc pas dans ce calcul. Notre hypothèse **H2** est motivée par cette estimation. Nous notons ce modèle $\mathbf{M}_{\mathrm{Id}}\mathbf{A}$.

D \ R	PR	MR	GR
DF	3.46	4.55	5.17
DI	3.46	4.55	5.17
DE	3.46	4.55	5.17

TABLE 5.1 – Récapitulatif des indices de difficultés de la tâche de sélection en fonction du rayon de demi-sphère **R** et de la densité de la scène **D**, pour le modèle $\mathbf{M}_{\mathrm{Id}}\mathbf{A}$. Pour un même rayon, les trois indices de difficultés sont égaux.

Nous pouvons également proposer une autre modèle, noté $\mathbf{M}_{\mathrm{Id}}\mathbf{B}$, en ne prenant plus en compte le rayon des cibles, mais leur taille effective W_e, comme le schématise la figure 5.8. Il s'agit de la taille des cellules de Voronoï associées aux cibles (voir la **Remarque** plus loin). La répartition homogène de nos cibles implique que cette taille est égale à la distance entre deux cibles adjacentes. Nous calculons alors l'indice de difficulté effectif $I_{de} = \log_2\left(1 + \frac{D}{W_e}\right)$. Avec les valeurs de rayon et de distance que nous avons choisies, nous observons que l'indice de difficulté effectif ne dépend plus du rayon de la demi-sphère, mais uniquement de la densité des cibles. Nous obtenons 1.52, 1.78 et 2.25 respectivement pour les trois densités. Ces valeurs sont récapitulées dans la table 5.2. Ce modèle indique que pour une densité donnée (c'est-à-dire un nombre de cibles donné), la difficulté de la tâche est constante quelle que soit la distance des cibles.

Remarque : Un diagramme de Voronoï est une décomposition particulière de l'espace déterminée par les distances à des objets de cet espace. Dans notre cas, un point de l'espace appartient à la cellule de Voronoï associée à une certaine cible si cette cible est la plus proche du point considérée.

D \ R	PR	MR	GR
DF	1.52	1.52	1.52
DI	1.78	1.78	1.78
DE	2.25	2.25	2.25

TABLE 5.2 – Récapitulatif des indices de difficultés de la tâche de sélection en fonction du rayon de demi-sphère **R** et de la densité de la scène **D**, pour le modèle $\mathbf{M}_{\mathrm{Id}}\mathbf{B}$. Pour une même densité, les trois indices de difficultés sont égaux.

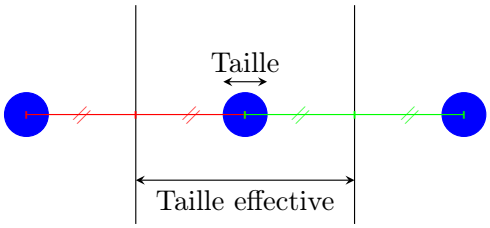


FIGURE 5.8 – Taille et taille effective des cibles pour l'évaluation de Bubble Bee. Cas où l'écart entre les cibles est constant.

5.4 Résultats

Une analyse de la variance (*ANOVA : ANalysis Of VAriance*) a été opérée sur les données récoltées pendant cette expérience, **D**, **T** et **R** étant les variables indépendantes, et le temps de décision et le niveau d'erreurs les variables dépendantes.

Lorsque l'ANOVA révèle un impact significatif d'une variable indépendante, nous effectuons une analyse post-hoc par le test de Tukey HSD (*Honestly Significant Difference*), afin d'identifier les différences significatives.

5.4.1 Temps de décision

La table 5.3 regroupe les temps de décision moyens, en fonction des valeurs des trois variables indépendantes.

L'analyse statistique révèle que la densité des cibles **D** a un impact significatif ($F_{2,\ 20} = 20.24$, $p < 0.05$) sur le temps de décision. L'analyse post-hoc indique que ce temps croît significativement en densité élevée, comparativement aux deux densités plus faibles. Ce résultat valide notre hypothèse **H1**. Il ne semble pas y avoir de différences significatives entre la densité intermédiaire et la densité faible, bien qu'une demi-seconde sépare les deux temps moyens.

Aucun impact significatif sur le temps de décision n'a été constaté pour la technique **T** ($F_{1,\ 10} = 4.378, p > 0.05$). L'information de direction semble donc être aussi intuitive pour le Bubble Bee que pour la flèche 3D. Ce résultat valide notre hypothèse **H3** et conforte notre motivation première, à savoir proposer une alternative efficace à la flèche 3D.

De même, le rayon de la demi-sphère **R** ne semble pas avoir d'impact significatif sur le temps de décision ($F_{2,\ 20} = 0.511, p > 0.05$). Ce résultat invalide notre hypothèse **H2**. Comme nous en avons discuté dans la section précédente, prendre en compte la taille effective des cibles, *i.e.* la taille de leur zone d'influence, est plus pertinent que prendre en compte leur taille « visuelle », pour estimer la difficulté de la tâche.

Variable	Valeur	Temps (en seconde)
D	**DF**	2.70
	DI	3.21
	DE	4.04
T	**BB**	3.64
	F3D	3.00
R	**PR**	3.22
	MR	3.31
	GR	3.42

TABLE 5.3 – Temps de décision moyens en fonction des variables indépendantes, obtenus lors de l'évaluation de Bubble Bee

5.4.2 Niveau d'erreur

La table 5.4 regroupe les niveaux d'erreur moyens, en fonction des valeurs des trois variables indépendantes.

L'analyse statistique montre un impact significatif de la densité **D** sur le niveau d'erreur ($F_{2,\ 20} = 193.4, p < 0.05$). L'analyse post-hoc relève, en comparant les densités deux-à-deux, que le niveau d'erreur croît significativement avec la densité. Ce résultat nous paraît cohérent avec le fait que la taille effective des cibles diminue lorsque la densité augmente. De la même manière, la distance des cibles **R** ne semble pas avoir d'impact significatif sur le niveau d'erreur.

En revanche, l'analyse statistique ne nous permet de conclure assurément si la technique **T** a un effet significatif sur le niveau d'erreur ($F_{1,\ 10} = 4.961, p = 0.0501$). Nous ne pouvons donc ni valider, ni invalider notre hypothèse **H4**. Toutefois, les niveaux d'erreurs moyens sont de $0,48$ pour la flèche 3D, et $0,56$ pour le Bubble Bee. Ce résultat indique que dans la très grande majorité des cas, les utilisateurs ont choisi la bonne cible ($NE = 0$) ou une cible adjacente ($NE = 1$).

Variable	Valeur	Niveau d'erreur
D	**DF**	0.2
	DI	0.46
	DE	0.89
T	**BB**	0.56
	F3D	0.48
R	**PR**	0.51
	MR	0.54
	GR	0.51

TABLE 5.4 – Niveaux d'erreur moyens en fonction des variables indépendantes, obtenus lors de l'évaluation de Bubble Bee

5.5 Discussion

En premier lieu, il est intéressant de noter que bien que le rayon des cibles soit constant durant toute l'expérimentation, le temps de décision et le niveau d'erreur dépendent de la taille des cellules de Voronoï associées aux cibles, *i.e.* leur taille effective. En effet, nous avions émis l'hypothèse que le temps de décision augmente si les cibles s'éloignent du Bubble Bee ou de la flèche 3D (**H2**). Or, l'analyse statistique invalide cette hypothèse. Pour une densité donnée, le temps de décision est similaire pour les trois distances proposées. Il semble donc que dans les conditions de cette expérience, le modèle $\mathbf{M}_{\mathrm{Id}}\mathbf{B}$ reflète plus correctement la difficulté de la tâche que le modèle $\mathbf{M}_{\mathrm{Id}}\mathbf{A}$.

Nous relevons également de ces résultats que le temps de décision est similaire pour les deux techniques. Cela suggère que comprendre l'information directionnelle fournie pour le Bubble Bee est aussi intuitif qu'avec la flèche 3D. De plus, l'erreur commise avec le Bubble Bee est presque aussi faible qu'avec la flèche 3D, et reste très satisfaisante. En effet, le niveau d'erreur moyen pour le Bubble Bee $(0,56)$ implique que l'utilisateur commet en moyenne une erreur angulaire de seulement 5 degrés.

Bien que l'analyse statistique ne permette pas de conclure quant à une différence significative entre les deux techniques sur le niveau d'erreur, il nous semble que la flèche 3D ait un léger avantage ($NE = 0.48$ en moyenne, contre $NE = 0.56$). Nous attribuons en partie cette différence au fait que les flèches sont omniprésentes dans le monde réel, en particulier sur les panneaux de signalisations et dans les rayons des magasins. Bien que ces indicateurs soient le plus souvent des schémas 2D, cette omniprésence nous permet d'associer très rapidement une flèche à sa sémantique « pointer une direction », même si cette flèche est en trois dimensions dans le cas d'un environnement immersif.

5.6 Conclusion

Dans ce chapitre, nous avons décrit une assistance graphique pour pointer une direction dans un environnement immersif, afin de proposer une alternative crédible à la flèche 3D, la technique la plus employée pour cette tâche. Notre motivation première pour concevoir cette aide était d'éviter l'utilisation de plusieurs outils simultanément en intégrant plusieurs fonctionnalités dans un seul outil. Le principe de notre aide consiste à afficher plusieurs anneaux alignés et orientés dans la même direction, pour définir un axe. Un dégradé de couleur permet de fixer une orientation pour cet axe. Nous avons appliqué ce principe, appelé Ring Concept, à une des techniques de sélection les plus connues et les plus performantes, le 3D Bubble Cursor. La technique résultante, appelée Bubble Bee, permet de localiser la cible la plus proche et aide l'utilisateur à planifier son geste de sélection.

Nous avons conçu un protocole expérimental permettant de quantifier et de comparer l'aide apportée par le Bubble Bee et par une flèche 3D. La tâche demandée aux participants de cette expérience était d'identifier la cible pointée par ces techniques. Les résultats montrent d'une part que si le taux d'erreur pour Bubble Bee est un peu plus élevé que pour la flèche 3D, la différence reste faible et les performances sont satisfaisantes dans les conditions de l'expérience. D'autre part, aucune différence significative n'a été observée pour les temps de décision, ce qui indique que les informations fournies par le Bubble Bee sont aussi compréhensibles que celles fournies par la flèche 3D. Nous attribuons en partie les différences observées entre les deux techniques au fait que les participants ont l'habitude de voir et d'interpréter les indications d'une flèche, en raison de leur omniprésence dans le monde réel. De plus, notre méthode a l'avantage de pouvoir être directement intégrée à de nombreux autres outils, ainsi que nous le faisons dans le chapitre suivant. Nous pensons donc que le Bubble Bee est une bonne alternative à l'utilisation combinée d'un 3D Bubble Cursor et d'une flèche 3D, en particulier dans les scènes où ces deux outils peuvent conduire à des problèmes d'occlusions.

Dans le cadre de notre travail, nous avons augmenté une technique de sélection par le Ring Concept. D'autres contextes applicatifs pourraient néanmoins appliquer ce principe. En effet, un outil suffisamment volumineux pour afficher des anneaux sur sa surface est un bon candidat pour être augmenté et donc enrichi d'une information directionnelle. La figure 5.9 illustre l'augmentation d'un objet non sphérique, une théière. Nous pouvons noter sur cet exemple que la direction donnée par les anneaux reste compréhensible malgré la forme non isotrope de l'objet. Cependant, l'orientation des anneaux peut être différente de l'orientation naturelle de l'objet (l'axe anse-goulot sur la figure). Il serait donc intéressant d'étudier l'influence de ce conflit sur la surcharge mentale et les performances de l'utilisateur.

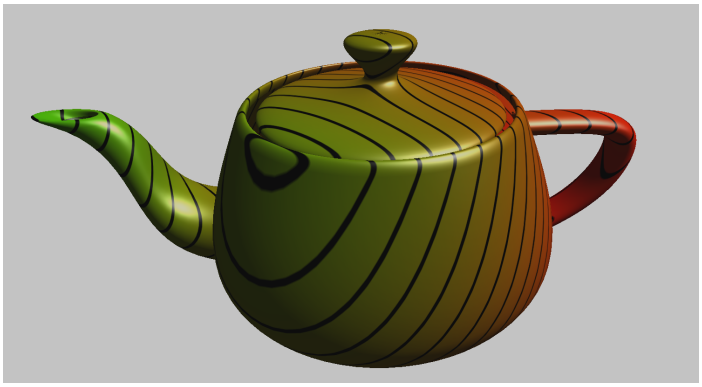


FIGURE 5.9 – La Teapot Bee. La théière est augmentée par le Ring Concept, fournissant par exemple la direction de la prochaine tasse de thé à remplir. L'orientation des anneaux peut être différente de l'orientation naturelle de l'objet, pouvant perturber l'utilisateur.

Dans certains jeux vidéo comme les FPS [1], des radars sont plus couramment employés que les flèches pour repérer la position d'ennemis cachés. L'arme du joueur pourrait être augmentée par le Ring Concept pour désigner par exemple l'ennemi le plus proche ou la sortie d'un labyrinthe, et remplacer ces radars pour gagner en espace visuel. Nous pourrions comparer les performances et l'expérience de jeu avec ces deux techniques.

Parmi les tâches courantes dans les environnements de réalité virtuelle, aligner plusieurs objets dans une même direction (par exemple des couverts sur une table), et diriger un objet vers un autre (par exemple un canon vers un château), sont deux tâches pouvant profiter du Ring Concept. En effet, la géométrie des objets considérés peut être complexe et son orientation naturelle difficile à percevoir, tandis que le Ring Concept est conçu pour définir sans ambigüité une direction. La figure 5.10 montre un objet quelconque entouré d'anneaux colorés, évoquant un Bubble Bee en fil de fer. Ici, le Ring Concept est utilisé comme widget d'orientation. Lorsque l'utilisateur déplace l'objet, les anneaux se déplacent avec lui. Aligner l'objet dans une certaine direction revient donc à aligner les anneaux, ce qui est une tâche plus simple et intuitive si l'objet a une géométrie complexe. Dans les différentes perspectives d'utilisation du Ring Concept que nous venons d'évoquer, afficher une flèche 3D plutôt qu'utiliser le Ring Concept est évidemment une solution souvent possible, mais qui ne nous satisfait pas pour des raisons d'occlusions et de perte d'espace visuel.

Dans le cas d'une scène dense, l'efficacité du 3D Bubble Cursor est fortement réduite et est comparable à un pointeur 3D basique. En effet, l'intérêt de cette technique réside dans sa capacité à sélectionner « à distance », en augmentant dynamiquement son rayon d'action. Or, dans une scène dense, il est nécessaire de s'approcher au plus près d'une

1. *First Person Shooter* : Jeu de tir en vue subjective (voir la figure 4.2).

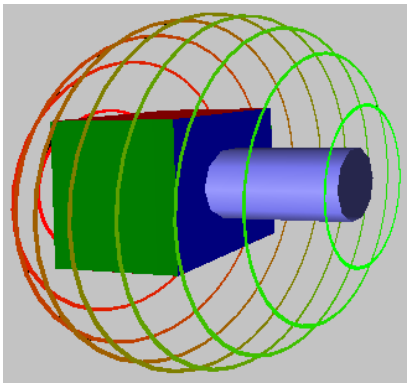


FIGURE 5.10 – Un widget d'orientation basé sur le Ring Concept, entourant un objet créé aléatoirement. Pour orienter correctement l'objet, l'utilisateur peut ne considérer que les anneaux, ce qui peut être plus intuitif si l'objet a une géométrie complexe.

cible pour que celle-ci soit la plus proche du pointeur, et donc pour qu'elle soit sélectionnable. Ce comportement se rapproche de celui d'un « simple » pointeur manipulé par l'utilisateur. Le gain en termes de temps de sélection est donc minime comparé à une scène moins dense. Dans le chapitre suivant, nous proposons une nouvelle technique de sélection appelée **Starfish**. Starfish est basée sur des contraintes virtuelles guidant le pointeur de l'utilisateur vers sa cible, permettant une sélection plus confortable. Le pointeur se déplace à l'intérieur de branches dont les extrémités sont placées sur les cibles à atteindre. Ces branches sont augmentées par le Ring Concept afin de fournir une meilleure perception de leur direction à l'utilisateur.

Chapitre 6

Starfish : Selection of TARgets from Implicit Surfaces with simulated Haptics

> Valkyrie One, this is Dragon.
> Target is in sight.
>
> Miles Quaritch, *Avatar*

Sommaire

6.1 Principe . . . 93
6.2 Introduction aux surfaces implicites . . . 96
6.3 Mise en œuvre . . . 99
6.3.1 Contrôle de Starfish . . . 99
6.3.2 Construction de la surface . . . 100
6.3.3 Contrainte du noyau . . . 110
6.3.4 Sélection de cible . . . 113
6.4 Évaluation de Starfish . . . 114
6.4.1 Matériel et participants . . . 114
6.4.2 Procédure . . . 115
6.4.3 Conception . . . 115
6.5 Résultats . . . 116
6.5.1 Temps de sélection . . . 116
6.5.2 Taux d'erreurs . . . 118
6.5.3 Questionnaire subjectif . . . 119
6.6 Discussion . . . 119
6.7 Conclusion . . . 121

Quelle que soit la technique employée, la sélection d'un objet dans une scène nécessite d'atteindre cet objet avec un effecteur, puis de valider ce choix. Dans le cas d'un contrôle direct, l'utilisateur déplace sa main pour toucher l'objet, et valide en pinçant deux doigts, par exemple. Dans le cas d'un rayon virtuel, l'utilisateur vise l'objet avec le rayon, en le manipulant à l'aide de son périphérique d'interaction. Enfin, dans le cas d'un pointeur comme le 3D Bubble Cursor ou le Bubble Bee, l'utilisateur déplace le pointeur afin qu'il touche l'objet. Dans les deux derniers cas, l'utilisateur peut généralement valider sa sélection à l'aide d'un bouton situé sur son périphérique, lorsque le rayon ou le pointeur touche l'objet à sélectionner. Néanmoins, en fonction de l'agencement de la scène et de la taille des cibles, la sélection peut nécessiter une précision et une concentration importante. En effet, outre les problèmes d'occlusions, la manipulation prolongée d'un périphérique d'interaction comme le flystick, donc sans support physique, engendre une fatigue se manifestant par des tremblements de la main, empêchant l'utilisateur de maintenir son rayon ou son pointeur sur la cible le temps de valider son choix. Notre objectif est de développer une nouvelle technique de sélection, basée sur un pointeur manipulé par l'utilisateur, mais guidé vers des cibles bien choisies à l'aide d'un support virtuel. L'utilisateur peut ainsi exécuter un geste moins précis, la technique l'aidant à atteindre sa cible, et à l'y maintenir jusqu'à et pendant l'étape de validation. Nous pensons que cette réduction de la précision requise entraîne une diminution du temps de sélection, et contribue à rendre la tâche de l'utilisateur plus confortable.

Dans ce chapitre, nous présentons notre nouvelle technique de sélection, ***Starfish***. Cette technique est basée sur la **contrainte du pointeur** à l'intérieur d'un volume, le guidant vers des cibles proches. Ces contraintes virtuelles prennent la forme d'une **surface implicite** déployant des **branches** vers des cibles bien choisies. Le pointeur est dans un premier temps libre de toute contrainte, et la surface implicite est reconstruite à chaque instant autour de ce pointeur. Lorsqu'une des branches **capture** la cible que souhaite atteindre l'utilisateur, celui-ci peut bloquer la surface pour guider son pointeur vers la cible. Nous commençons dans ce chapitre par détailler le processus de réflexion nous menant au développement de Starfish, ainsi que son implémentation. Nous faisons également un bref rappel sur le concept de surface implicite, afin d'expliciter les notations que nous utilisons par la suite. Nous évaluons ensuite les performances de Starfish dans une tâche de sélection à l'aide d'un protocole expérimental comparant notre technique à une technique de rayon. Pour finir, nous présentons et discutons les résultats de cette expérience.

6.1 Principe

Parmi les techniques de sélection que nous avons évoquées dans le chapitre 4, deux aspects nous paraissent intéressants à développer. D'une part, certaines techniques proposent une présélection d'un sous-ensemble de cibles et offrent un menu à l'utilisateur afin qu'il finalise son choix. Le SQUAD, par exemple, permet une présélection à l'aide

d'un rayon, et un raffinement progressif à l'aide d'un quad-menu (menu découpant la zone d'affichage en 4 parties) [Kopper 11]. Bien qu'une sélection rapide soit possible quelle que soit la densité de la scène (seuls $log_4(n)$ clics sont nécessaires, où n est le nombre d'objets présélectionnés), l'utilisateur perd le contexte entre les objets et la scène. La technique Expand propose une solution à ce problème en affichant une animation entre l'objet dans la scène, et sa représentation dans le menu [Cashion 12], mais nous pensons que laisser les objets pré-sélectionnés « en place » éviterait de perturber l'utilisateur, et permettrait d'améliorer sa compréhension de la scène. Nous nous limitons ici à la sélection des cibles proches, *i.e.* à portée du pointeur $3D$.

D'autre part, nous estimons que contraindre le pointeur de l'utilisateur dans un certain volume est une solution élégante pour guider son geste durant le processus de sélection. Cette solution est principalement employée dans les menus 3D, avec des contraintes soit uniquement visuelles comme le C^3 [Grosjean 01], soit en ajoutant une composante haptique comme HardBorders [Essert-Villard 09]. Par ailleurs, ces deux techniques proposent des menus dont les items sont placés selon un schéma simple, comme un cube ou un polygone. Nous souhaitons adapter cette approche à un cas plus générique, où nous n'avons pas d'*a priori* sur la disposition des cibles. Dans cette approche, des guides s'adapteraient dynamiquement à la position dans l'espace du pointeur et des cibles, comme l'illustre la figure 6.1. Se pose alors la question de la mise en œuvre de ces guides. Quelle forme devrait avoir le volume pour guider correctement le pointeur ?

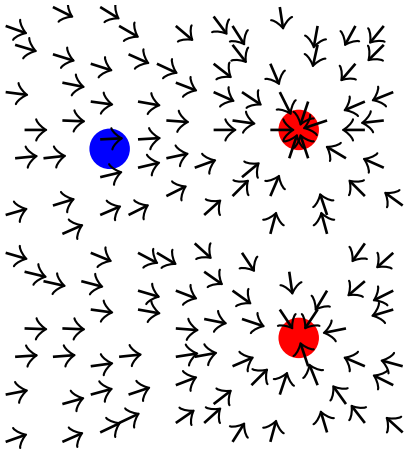


FIGURE 6.1 – Illustration du comportement souhaité pour le pointeur (en bleu) selon sa position dans l'espace. Des contraintes se mettent en place pour guider le pointeur vers l'une des cibles (en rouge).

Notre idée est de créer des « branches » dans lesquelles le pointeur manipulé par l'utilisateur peut s'engouffrer, glisser sur les parois, et s'arrêter sur une cible. Cette approche

nécessite autant de branches que de cibles pré-sélectionnées. La forme la plus naturelle pour ces branches est le cône. En effet, le pointeur peut aisément entrer dans le cône par sa base élargie, et glisser sur les parois pour atteindre précisément la cible choisie au sommet du cône. La figure 6.2 (gauche) propose un schéma 2D de cette approche. Afin de guider efficacement l'utilisateur, nous souhaitons lui proposer une surface continue et fermée, formant un volume duquel le pointeur ne pourra pas sortir. Une solution possible pour fermer la surface à partir des cônes est de simplement les étendre, comme l'illustre la figure 6.2 (centre). Néanmoins, cette solution n'est pas satisfaisante, car les ouvertures accordées pour chaque cible ne sont pas égales. En effet, plus une cible est lointaine, plus son ouverture est grande. À l'inverse, les cibles trop proches sont difficiles à sélectionner, car entrer dans la branche correspondante demande une précision très importante.

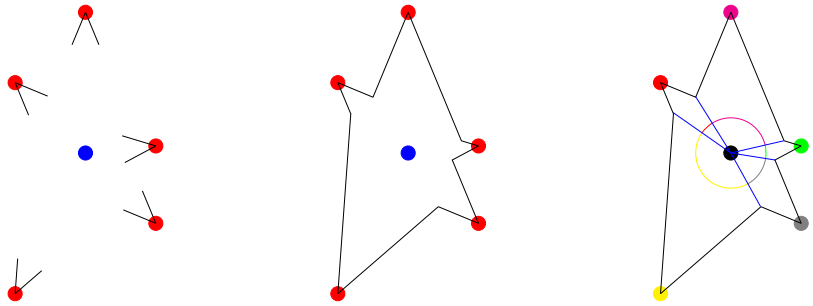


FIGURE 6.2 – Illustration 2D des guides basés sur des cônes. Les sommets des cônes sont placés sur les cibles (en rouge). Les cônes sont orientés vers le pointeur (en bleu). **Gauche :** De petits cônes sont créés sur les cibles proches du pointeur. **Centre :** Les cônes sont étendus pour proposer une surface continue, et garder le pointeur à l'intérieur de ce volume. **Droite :** Cette solution n'est pas satisfaisante. Les ouvertures vers les cibles n'ont pas toutes la même taille, rendant difficile la sélection des cibles proches.

La solution que nous proposons est basée sur l'union de plusieurs branches générées par une surface implicite. Nous appliquons une fonction de potentiel bien choisie à un squelette composé de segments reliant le pointeur 3D de l'utilisateur à des cibles proches. Chaque segment génère un champ de potentiel dont la forme évoque une branche, se terminant au niveau de la cible. L'union de ces branches forme une surface fermée, à l'intérieur de laquelle le pointeur est virtuellement contraint, et rappelant la forme d'une étoile de mer. Nous appelons *Starfish* (pour *Selection of TARgets From Implicit Surfaces with simulated Haptics*) cette nouvelle technique. Dans la suite de ce document, nous appelons le pointeur 3D le *noyau* de Starfish.

Il est évident que nous ne pouvons pas, pour des raisons d'occlusions, créer une branche pour chacune des cibles présentes dans la scène. Il est donc nécessaire de choisir un sous-ensemble de cibles qui seront capturées par les branches. Ce choix est réévalué à chaque instant afin de prendre en compte le déplacement de l'utilisateur dans la scène. Les cibles les plus à même d'être sélectionnées, selon un certain critère, sont alors captu-

rées par les branches. La sélection d'une cible particulière suit donc un processus en deux phases. L'utilisateur déplace dans une première phase le noyau afin que la cible choisie soit capturée par une branche. Dans une seconde phase, l'utilisateur bloque le volume obtenu par les branches, et déplace le noyau à l'intérieur de ce volume, jusqu'à atteindre la cible à sélectionner. Ce processus rappelle les techniques de type raffinement progressif comme SQUAD. Le nombre de cibles potentielles est d'abord réduit par une présélection, puis les cibles restantes sont présentées dans un menu. Ici, Starfish est l'équivalent d'un menu 3D dont les items dépendent de la position du noyau.

Afin de mieux comprendre la mise en œuvre de notre solution, la section suivante est consacrée à une brève introduction au concept de surface implicite, que nous limitons volontairement aux notions qui nous sont utiles dans la suite, et que nous adaptons à notre contexte applicatif.

6.2 Introduction aux surfaces implicites

En infographie, la méthode la plus courante pour modéliser une surface est d'utiliser des équations paramétriques de cette surface pour en définir des points, qui sont ensuite reliés pour former un maillage polygonal. Les surfaces implicites sont une alternative à cette représentation. Une surface implicite S [Ricci 73, Blinn 82] est l'ensemble des points $X = (x, y, z)$ de l'espace vérifiant $f(X) = \alpha$, pour une certaine fonction $f : \mathbb{R}^3 \to \mathbb{R}$, et un certain réel α.

Nous pouvons voir la fonction f comme un champ de potentiel. Nous disons alors que S est la surface isopotentielle de niveau α de la fonction f. Nous pouvons placer chaque point X de l'espace par rapport à la surface, en évaluant la fonction f en ce point. Si $f(X) > \alpha$, nous disons que X est à l'intérieur de la surface. Si $f(X) < \alpha$, nous disons qu'il est à l'extérieur. La figure 6.3 représente plusieurs surfaces isopotentielles de la fonction $f(x, y) = x^2 + y^2$. Pour une courbe donnée, f prend la même valeur pour tous les points de cette courbe.

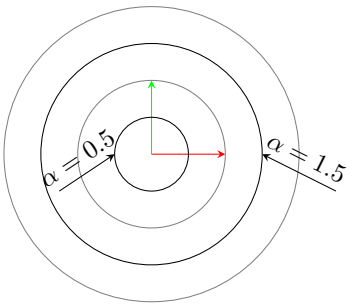


FIGURE 6.3 – Surfaces isopotentielles de la fonction $f(x, y) = x^2 + y^2$.

Il existe plusieurs approches pour représenter une surface implicite. L'approche algébrique utilise un polynôme pour fonction f. Ce polynôme est généralement de degré faible afin de réduire le temps de calculs. Les superquadriques sont une généralisation de cette méthode [Barr 81]. Une autre approche, plus répandue, est basée sur la notion de squelette [Galin 97]. Dans ce contexte, un squelette est un ensemble de primitives générant un champ de potentiel. Le potentiel d'un point de l'espace est obtenu en mélangeant les contributions de tous les champs en ce point. Nous parlons également de *blobs* pour désigner ces primitives.

Le blob le plus simple est le point, générant un champ radial. Ce champ est maximal au niveau du point, et décroît avec la distance à ce point. La figure 6.4 illustre la forme d'un champ de potentiel généré par deux blobs ponctuels. Les champs respectifs des deux blobs se mélangent lorsque les points se rapprochent. Pour chaque point de l'espace (ou du plan, sur la figure), nous calculons sa distance euclidienne avec le centre du blob. À une distance, nous associons un potentiel. Une fonction de potentielle est décroissante en fonction de la distance au squelette et positive. La fonction suivante est un exemple classique de fonction de potentiel :

$$p(d) = \begin{cases} 0 & \text{si } d \geqslant D_{\max}, \\ \left(1 - \dfrac{d^2}{{D_{\max}}^2}\right)^2 & \text{sinon.} \end{cases} \tag{6.1}$$

où $D_{\max}$ est une distance maximale, préalablement fixée, au-delà de laquelle le champ généré est nul. Le champ généré vaut 1 au centre du blob, et diminue radialement jusqu'à atteindre une valeur nulle à une distance $D_{\max}$. Nous pouvons noter que les surfaces isopotentielles sont également des surfaces isodistances, car le potentiel d'un point ne dépend que de sa distance avec le centre du blob.

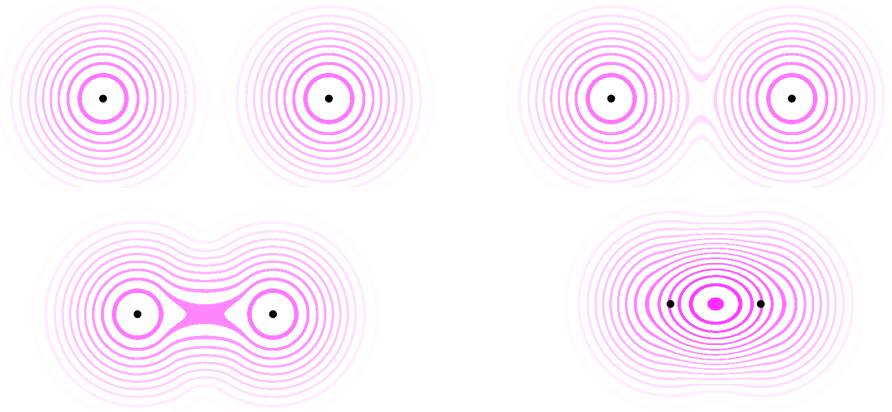


FIGURE 6.4 – Mélange des champs de potentiels radiaux, en fonction de la distance entre les deux primitives.

Un autre blob souvent utilisé est le segment, comme illustré en figure 6.5. La même fonction de potentiel que pour le blob ponctuel peut être utilisée. La distance d'un point X de l'espace au blob segment est égale à la distance minimale entre ce point et les points du segment. Cette quantité peut être calculée de la manière suivante. Appelons S_1 et S_2 les deux extrémités du segment. Soit $\overrightarrow{u}$ un vecteur unitaire de ce segment :

$$\overrightarrow{u} = \frac{\overrightarrow{S_1S_2}}{\|S_1S_2\|}. \tag{6.2}$$

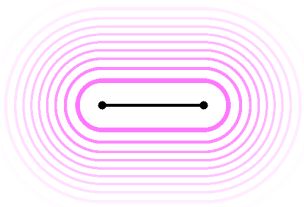


FIGURE 6.5 – Champ de potentiel généré par un segment. La distance d'un point au segment est égale à la distance entre ce point, et le point le plus proche du segment.

La quantité $p(X) = \overrightarrow{S_1X} \cdot \overrightarrow{u}$, projection du point X sur la droite (S_1S_2) (voir figure 6.6), nous informe de la position de X par rapport aux points S_1 et S_2, et nous permet de calculer sa distance au segment $[S_1S_2]$:

- Si $p(X) \leqslant 0$, alors le projeté de X sur la droite est à l'extérieur du segment. La distance entre X et le segment est égale à la distance euclidienne entre X et S_1.
- De même, si $p(X) \geqslant \|\overrightarrow{S_1S_2}\|$, alors la distance entre X et le segment est égale à la distance euclidienne entre X et S_2.
- Sinon, le projeté de X est à l'intérieur du segment. Il suffit alors de calculer la distance euclidienne entre X et son projeté.

La figure 6.7 illustre ces trois cas de figure.

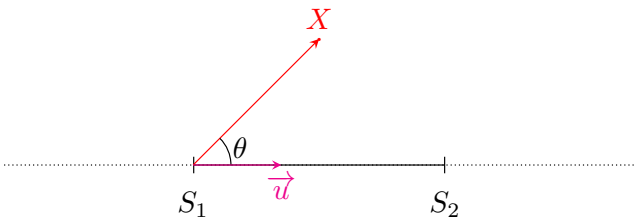


FIGURE 6.6 – Calcul de la quantité $p(X)$, projection de X sur un segment.

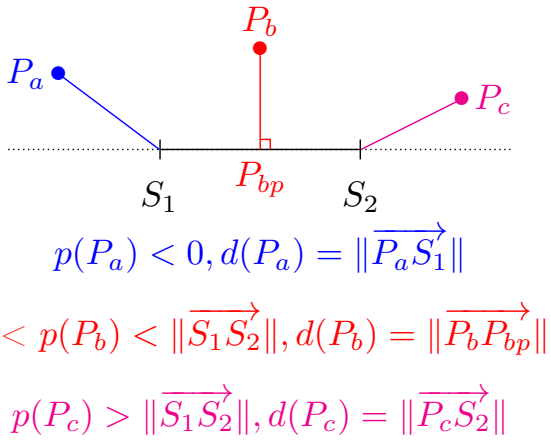


FIGURE 6.7 – Calcul de distance d'un point à un segment $[S_1S_2]$ en fonction de la position du projeté de ce point sur l'axe du segment. Le projeté du point P_a (resp. P_c) est à l'extérieur du segment. La distance de P_a (resp. P_c) au segment est égale à sa distance euclidienne au point S_1 (resp. S_2). Le projeté P_bp du point P_b est à l'intérieur du segment. La distance de P_b au segment est égale à sa distance euclidienne au point P_bp.

6.3 Mise en œuvre

Dans cette section, nous détaillons la mise en œuvre des différents aspects de Starfish, en particulier le contrôle global de notre technique, la construction dynamique de la surface implicite et le déplacement contraint du pointeur sur cette surface.

6.3.1 Contrôle de Starfish

Notre technique nécessite un dispositif d'interaction afin de déplacer le noyau de Starfish dans l'espace, ainsi que deux boutons, le bouton **Move** et le bouton **Select**. Tant

qu'aucun bouton n'est pressé, le noyau est immobile. Cet état, appelé mode ***Idle***, permet à l'utilisateur de relâcher son bras sans déplacer Starfish. Tant que le bouton **Move** est pressé, la position du noyau est associée à la position du dispositif. Nous appelons cet état le mode ***Move***. À chaque instant, la surface de Starfish est reconstruite en se basant sur la nouvelle position de son noyau, comme nous l'expliquons en 6.3.2. Relâcher le bouton **Move** active le mode ***Idle*** à nouveau.

Tant que le bouton **Select** est pressé, la position du noyau de Starfish est également associée à la position du dispositif, mais la surface n'est pas reconstruite. De plus, le noyau est contraint visuellement à l'intérieur du volume formé par la surface. Nous décrivons en 6.3.3 l'algorithme permettant de gérer le comportement du noyau sur la surface. Nous appelons cet état, le mode ***Select***. Lorsque l'utilisateur relâche le bouton **Select**, la technique vérifie en premier lieu si une cible est sélectionnée, comme expliqué en 6.3.4. Si c'est le cas, l'utilisateur peut interagir avec cette cible, en fonction de l'application. Dans le cas contraire, Starfish retourne en mode ***Idle***. Une cible ne peut être sélectionnée que durant le mode ***Select***.

La figure 6.8 est une illustration de l'utilisation de Starfish. La position du noyau est liée à celle de la main de l'utilisateur manipulant le périphérique.

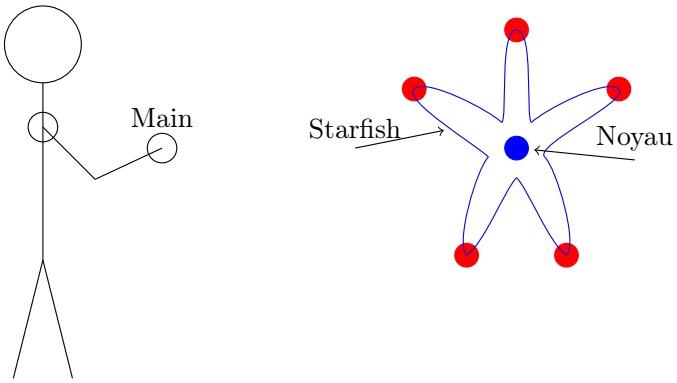


FIGURE 6.8 – Illustration du contrôle de Starfish. La position du noyau de Starfish est liée à la position du périphérique d'interaction manipulé par l'utilisateur. Les branches de Starfish sont construites autour de son noyau, et capturent chacune une cible (en rouge).

6.3.2 Construction de la surface

Nous rappelons que l'objectif de Starfish est d'aider localement l'utilisateur à sélectionner une cible. Cette aide se traduit par une présélection d'un sous-ensemble de cibles, et par des contraintes spatiales sur le pointeur de l'utilisateur, ou *noyau* de Starfish, placées de telle manière que seules les cibles présélectionnées puissent être atteintes par le pointeur. Dans ce but, nous construisons une surface fermée « capturant » les cibles pré-

sélectionnées, et contraignons le pointeur à l'intérieur du volume formé par la surface. Nous faisons l'hypothèse que la forme d'une étoile de mer (*i.e.* un ensemble de branches autour d'un noyau central) est adéquat. En effet, les contraintes permettent à l'utilisateur d'atteindre la cible choisie, sans accomplir de geste précis, comme l'illustre la figure 6.9. De plus, l'aspect visuel des branches informe l'utilisateur de la disposition de la scène, et dans quelles directions il doit déplacer le noyau pour atteindre les cibles capturées.

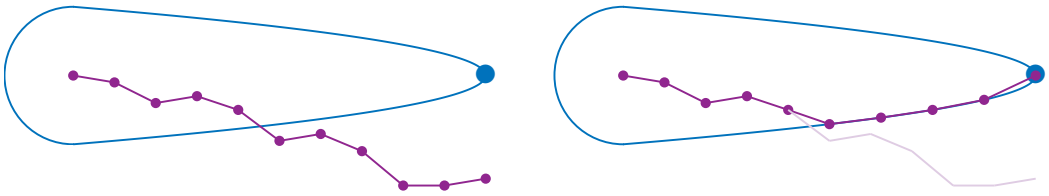


FIGURE 6.9 – Déplacement du noyau de Starfish sur la surface d'une branche. **Gauche :** Exemple de mouvement sans appliquer de contraintes sur le noyau. Le noyau suit le mouvement de l'utilisateur. Atteindre la cible est difficile, du fait de la grande précision nécessaire. **Droite :** Le noyau suit le mouvement de l'utilisateur, tout en restant contraint à l'intérieur de la branche. La cible peut être facilement atteinte même si l'utilisateur n'est pas précis.

Nous pouvons maintenant nous poser deux questions :
- Quels sont les critères pour déterminer les cibles à présélectionner ?
- Comment construire et afficher la surface de Starfish ?

Présélectionner les cibles À chaque instant lors de son utilisation, Starfish doit déterminer quelles cibles peuvent être sélectionnées. Ces cibles définissent un squelette sur lequel la surface est construite, *i.e.* un ensemble de segments entre le noyau, que nous notons N_0, et les cibles C_i. À cette fin, nous suivons l'algorithme suivant, basé sur des filtres successifs intuitifs :

1. **Initialisation :** Nous commençons par considérer toutes les cibles de la scène.
2. **Filtre de distance :** Starfish est une aide locale à la sélection. Toutes les cibles capturées par Starfish devraient être à portée de l'utilisateur, et atteignables en un seul geste. Nous rejetons donc toutes les cibles dont la distance euclidienne avec le noyau est supérieure à un certain paramètre $R_{\max}$.
3. **Filtre d'angle :** Si l'écart angulaire entre deux branches est trop petit, il peut être difficile de s'engouffrer dans la branche choisie sans faire d'erreurs. De plus, comme nous le verrons plus loin, si deux cibles sont alignées avec le noyau, le champ de potentiel résultant ne permettra pas de s'arrêter au niveau de la cible la plus proche. Afin de pallier ce problème, nous vérifions que l'écart angulaire entre

deux cibles n'est pas plus petit qu'un certain paramètre $\theta_{\min}$. Le cas échéant, nous rejetons la cible la plus éloignée du noyau. Afin de garder un résultat cohérent si le noyau est immobile, l'ordre de vérification est toujours le même. En pratique, nous trions au préalable les cibles par leur distance croissante avec le noyau. La figure 6.10 illustre le fonctionnement de ce filtre.

4. **Filtre de quantité :** Plus le nombre de branches est élevé, plus la visualisation de la surface de Starfish peut être confuse et obstruer la scène. S'engouffrer dans une branche est donc plus difficile ; nous préférons employer un petit nombre de branches. Parmi les cibles restantes, nous ne retenons que les $\mathfrak{N}_{\max}$ plus proches du noyau.

Après l'application de ces filtres, l'objet que l'utilisateur souhaite sélectionner peut avoir été rejetée. Néanmoins, la surface est dynamiquement recréée à chaque instant, il est alors simple et intuitif d'ajuster la position du noyau afin de capturer la bonne cible. L'ensemble des segments entre le noyau et les cibles retenues est appelé le *squelette* de Starfish.

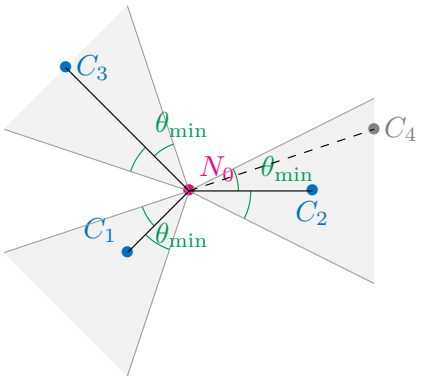


FIGURE 6.10 – Filtre d'angle pour le squelette de Starfish. C_1 et C_3 n'entrent en conflit avec aucune autre cible. En revanche, l'angle $\widehat{C_2N_0C_4}$ est plus petit que $\theta_{\min}$. La distance N_0C_2 étant plus petite que la distance N_0C_4, nous rejetons la cible C_4.

Surface implicite Chaque segment relie le noyau à une des cibles présélectionnées. Nous souhaitons que la surface générée par un tel segment permette au noyau de suivre la surface pour atteindre la cible, puis de se caler à son niveau. La figure 6.11B montre la forme que nous souhaitons obtenir. Les champs générés par les différents segments sont ensuite mélangés pour former le champ de potentiel du squelette. La surface de Starfish peut alors être vue comme l'ensemble des points de l'espace partageant une même valeur de potentiel pour ce champ. Cette valeur, que nous notons ψ_0, doit être choisie afin que la surface isopotentielle traverse toutes les cibles capturées. ψ_0 doit également être non nulle, afin de pouvoir déterminer le gradient du champ de potentiel au niveau de la surface. Cette propriété sera utilisée pour contraindre le noyau, comme nous l'expliquerons

en 6.3.3. Pour créer une telle surface implicite, nous avons besoin :

- d'une fonction de potentiel Ψ, et
- d'une fonction de distance au squelette $\mathfrak{d}$.

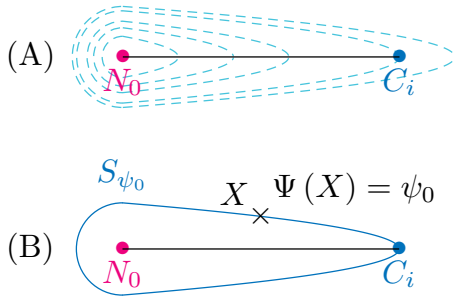


FIGURE 6.11 – Champ de potentiel généré par un squelette composé d'un unique segment. **(A) :** Surfaces isopotentielles générées par le segment $[N_0C_i]$. **(B) :** Cette surface est l'ensemble des points pour lesquels le potentiel est égal à ψ_0. La surface passe par la cible C_i, et entoure le noyau N_0. Cette forme permet de glisser sur la surface pour atteindre la cible.

La fonction $\Psi : \mathbb{R}^3 \longrightarrow \mathbb{R}$ indique le potentiel d'un point X en fonction de sa distance au squelette. Afin de s'assurer que la surface soit continue et fermée, Ψ doit être décroissante en fonction de la distance au squelette (au sens de $\mathfrak{d}$), positive, continue. Comme nous le verrons en 6.3.3, Ψ doit également être différentiable. En effet, nous exploitons la normale à la surface pour contraindre notre pointeur. De nombreuses fonctions ont été proposées dans la littérature [Blinn 82, Nishimura 85, Tsingos 95, Guiard 00]. Si une comparaison fine des fonctions possibles est au-delà du cadre de ce document, nous suggérons, pour des raisons pratiques de coût de calculs, d'utiliser la fonction suivante, polynomiale et à support fini :

$$\Psi(X) = \begin{cases} 0 & \text{si } \mathfrak{d}(X) \geqslant D_{\max}, \\ \left(1 - \dfrac{\mathfrak{d}(X)^2}{{D_{\max}}^2}\right)^2 & \text{sinon.} \end{cases} \tag{6.3}$$

$D_{\max}$ est un paramètre dont la valeur doit être déterminée au préalable, et représente la distance maximale (au sens de la fonction $\mathfrak{d}$) à laquelle le squelette génère un champ non nul. Ce paramètre fixe donc l'épaisseur des branches. Le choix de cette fonction Ψ implique que tous les points d'une même surface isopotentielle ont la même distance au squelette. Pour obtenir la forme d'une branche, une distance euclidienne n'est donc pas adéquate.

La fonction de distance au squelette $\mathfrak{d}$ doit être définie de telle sorte que la surface isopotentielle S_{ψ_0}, dont tous les points ont un potentiel $\Psi = \psi_0$, traverse toutes les cibles présélectionnées. Nous simplifions ce problème en définissant une fonction de distance $\mathfrak{d}_i$ pour chaque segment $[N_0 C_i]$ du squelette. Ces fonctions sont ensuite mélangées pour obtenir la fonction de distance au squelette complet :

$$\mathfrak{d}(X) = \min_{i=1\ldots n} \left(\mathfrak{d}_i(X)\right), \tag{6.4}$$

où n est le nombre de segments dans le squelette, avec évidemment $1 \leqslant n \leqslant \mathfrak{N}_{\max}$. L'apparence que nous souhaitons obtenir pour la surface S_{ψ_0} pour un squelette composé d'un unique segment, est illustrée en figure 6.12. Chaque point X de cette surface, en incluant C_i, a un potentiel $\Psi(X) = \psi_0$. En manipulant l'équation (6.3), nous obtenons alors la distance de X au segment :

$$\mathfrak{d}_i(X)_{X \in S_{\psi_0}} = D_{\max} \cdot \sqrt{1 - \sqrt{\psi_0}} = \mathfrak{d}_{\psi_0} \tag{6.5}$$

$$\mathfrak{d}_i(C_i) = \mathfrak{d}_{\psi_0} \tag{6.6}$$

Nous pouvons bien sûr noter que $\mathfrak{d}_{\psi_0} < D_{\max}$. Le noyau de Starfish, N_0, est à l'intersection de toutes les branches, et doit donc avoir une distance nulle à chaque segment :

$$\forall i \in [\![1; n]\!], \mathfrak{d}_i(N_0) = 0 \tag{6.7}$$

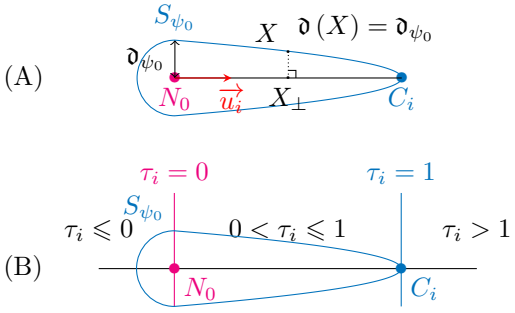


FIGURE 6.12 – Surface S_{ψ_0} de Starfish. **(A) :** Chaque point de la surface a une distance $\mathfrak{d}_{\psi_0}$ au segment, au sens de la fonction $\mathfrak{d}$. La distance d'un point de l'espace au segment $[N_0 C_i]$ ne dépend que de sa coordonnée le long de l'axe $(N_0 C_i)$. **(B) :** Représentation des valeurs prises par la fonction τ_i, avec $\tau_i(X)$ entre 0 et 1 si $X_\perp$, projeté de X sur l'axe $(N_0 C_i)$, est à l'intérieur du segment $[N_0 C_i]$.

Le champ de potentiel généré par le segment $[N_0C_i]$ admet une symétrie par rotation le long de ce segment. Pour tout point X de l'espace, la valeur $\mathfrak{d}_i(X)$ ne dépend donc que de la coordonnée de X sur l'axe (N_0C_i). Nous définissons cette coordonnée, $\tau_i(X)$, de la manière suivante :

$$\tau_i(X) = \frac{\overrightarrow{N_0X_\perp} \cdot \overrightarrow{u_i}}{\|\overrightarrow{N_0C_i}\|}, \tag{6.8}$$

où $\overrightarrow{u_i}$ est un vecteur directeur unitaire de l'axe (N_0C_i), et $X_\perp$ est le projeté orthogonal de X sur cet axe. Cette fonction vérifie les propriétés suivantes, illustrées en figure 6.12B :

- $\tau_i(X) = 0$ si $X_\perp = N_0$
- $\tau_i(X) = 1$ si $X_\perp = C_i$

À partir de cette définition et de la forme recherchée de la surface S_{ψ_0}, nous pouvons déterminer deux zones d'intérêt pour la fonction de distance $\mathfrak{d}$. Lorsque $\tau_i < 0$, la surface a la forme d'une demi-sphère : tous les points d'une même surface isodistance ont la même distance euclidienne à N_0. Lorsque $\tau_i \geqslant 0$, nous souhaitons une branche, dont le rayon est maximal pour $\tau_i = 0$ et diminue progressivement. La branche correspondant à la surface de potentiel ψ_0 a pour extrémité le point C_i, avec $\tau_i = 1$. Ces deux zones forment deux demi-espaces complémentaires dans $\mathbb{R}^3$, partagés par le plan d'équation $\tau_i(X) = 0$. Nous proposons la formulation suivante pour la fonction $\mathfrak{d}_i$:

$$\mathfrak{d}_i(X) = \begin{cases} \dfrac{(N_0X)^2}{\mathfrak{d}_{\psi_0}} & \text{si } \tau_i(X) < 0, \\ \dfrac{(XX_\perp)^2}{\mathfrak{d}_{\psi_0}} + \mathfrak{d}_{\psi_0} \cdot \tau_i(X) & \text{sinon.} \end{cases} \tag{6.9}$$

Vérifions que cette fonction vérifie les propriétés que nous nous sommes fixées. $\mathfrak{d}_i$ est clairement positive. En N_0, le noyau du squelette, nous avons $\tau_i(N_0) = 0$, et $N_{0\perp} = N_0$, donc :

$$\begin{aligned} \mathfrak{d}_i(N_0) &= \frac{(N_0N_{0\perp})^2}{\mathfrak{d}_{\psi_0}} + \mathfrak{d}_{\psi_0} \cdot \tau_i(N_0) \\ &= 0 \end{aligned}$$

En C_i, nous avons $\tau_i(N_0) = 1$ et $C_{i\perp} = C_i$, donc :

$$\begin{aligned} \mathfrak{d}_i(C_i) &= \frac{(C_iC_{i\perp})^2}{\mathfrak{d}_{\psi_0}} + \mathfrak{d}_{\psi_0} \cdot \tau_i(C_i) \\ &= \mathfrak{d}_{\psi_0} \end{aligned}$$

Nous vérifions avec l'équation (6.3) que nous avons les bonnes valeurs de potentiel en ces deux points :

$$\begin{aligned}\Psi\left(N_0\right) &= \left(1 - \frac{\mathfrak{d}\left(N_0\right)^2}{{D_{\max}}^2}\right)^2 \\ \Psi\left(N_0\right) &= 1\end{aligned} \tag{6.10}$$

$$\begin{aligned}\Psi\left(C_i\right) &= \left(1 - \frac{\mathfrak{d}\left(C_i\right)^2}{{D_{\max}}^2}\right)^2 \\ &= \left(1 - \frac{\mathfrak{d}_{\psi_0}^2}{{D_{\max}}^2}\right)^2 \\ &= \left(1 - \frac{D_{\max}^2 \cdot \left(1 - \sqrt{\psi_0}\right)}{{D_{\max}}^2}\right)^2 \\ \Psi\left(C_i\right) &= \psi_0\end{aligned} \tag{6.11}$$

Nous vérifions également la continuité de $\mathfrak{d}_i$. Dans le demi-espace d'équation $\tau_i\left(X\right) < 0$, $\mathfrak{d}$ correspond à la distance euclidienne avec le point N_0 (à un facteur multiplicateur $\frac{1}{\mathfrak{d}_{\psi_0}}$ près), et est donc continue. Dans le demi-espace d'équation $\tau_i\left(X\right) \geqslant 0$, la fonction $\mathfrak{d}_i$ est continue en tant que somme de fonctions continues. Il nous reste à montrer la continuité entre les deux demi-espaces. Soit X_0 un point tel que $\tau_i\left(X_0\right) = 0$. Son projeté sur l'axe (N_0C_i) est N_0. Nous avons donc :

$$\mathfrak{d}_i\left(X_0\right) = \frac{\left(N_0X_0\right)^2}{\mathfrak{d}_{\psi_0}} \tag{6.12}$$

La continuité de $\mathfrak{d}_i$ en X_0 est alors aisée à démontrer par continuité de la distance euclidienne :

$$\lim_{\substack{X \to X_0 \\ \tau_i(X) < 0}} \mathfrak{d}_i\left(X\right) = \frac{\left(N_0X_0\right)^2}{\mathfrak{d}_{\psi_0}} \tag{6.13}$$

Lorsque le squelette est composé de plus d'une branche, la distance au squelette d'un point est défini comme sa plus petite distance à une des branches, comme expliqué par l'équation (6.4). Les champs de potentiel se mélangent pour former des surfaces fermées et continues. La figure 6.13 illustre la surface S_{ψ_0} dans le cas d'un squelette composé de 4 segments. Cette surface traverse toutes les cibles.

De l'utilisation du terme *distance* Nous avons souvent fait référence à la fonction $\mathfrak{d}_i$ comme étant une fonction de distance à un segment, et à la fonction $\mathfrak{d}$ comme une

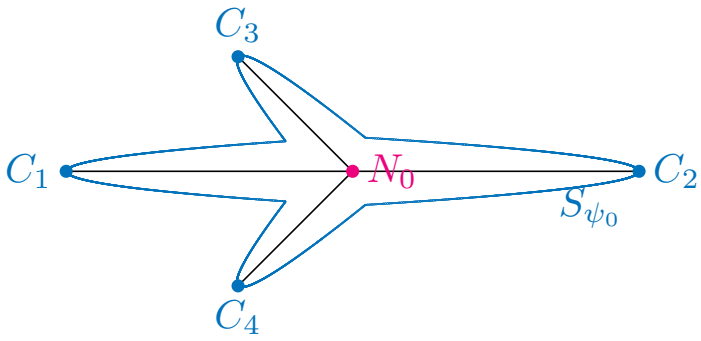


FIGURE 6.13 – Surface S_{ψ_0} générée par un squelette à 4 segments.

fonction de distance à un squelette. Ce terme « distance » est employé pour fournir une approche intuitive des définitions que l'on donne, et n'est pas une distance au sens mathématique, c'est-à-dire une application $E \times E \longrightarrow \mathbb{R}^+$, avec E un espace vectoriel, vérifiant les propriétés suivantes :

- $d(x,y) = 0$ si et seulement si $x = y$ (séparation)
- $d(x,y) = d(y,x)$ pour tout $x, y \in E$ (symétrie)
- $d(x,z) \leqslant d(x,y) + d(y,z)$ pour tout $x, y, z \in E$ (inégalité triangulaire)

Nous rappelons également que la distance d'un point à un segment est définie comme le minimum des distances entre ce point et chaque point du segment. De manière générale, si A est une partie non vide de E ($A \in \mathcal{P}(E)$, $A \neq \emptyset$), et x un élément de E, la distance de x à A est définie de la manière suivante :

$$d(x,A) = \inf \{d(x,y), y \in A\}$$

Affichage de la surface En manipulant Starfish en mode ***Move***, l'utilisateur doit s'assurer que la cible qu'il souhaite sélectionner est capturée par une des branches. Il est donc nécessaire d'afficher la surface S_{ψ_0}. La reconstruction d'une telle surface à partir du champ de potentiel est l'objet de plusieurs études [Hoppe 92, Galin 00, Ferley 00, Akkouche 01] qui sont pour la plupart des améliorations des algorithmes Marching Cube [Lorensen 87] ou Marching Triangles [Hilton 96].

Une mauvaise perception de la position des objets de la scène est souvent la source de mauvaises performances. Sur les trois dimensions, la profondeur est la plus mal interprétée [Fiorentino 03]. Des indices visuels, comme des ombres, sont donc couramment employés pour pallier ce problème. Dans notre contexte, il s'avère qu'il est difficile de

percevoir dans quelle direction une branche pointe, si elle est affichée avec une couleur uniforme, comme le montre la figure 6.14. Déplacer le noyau dans ce type de volume est donc sujet à de nombreuses erreurs.

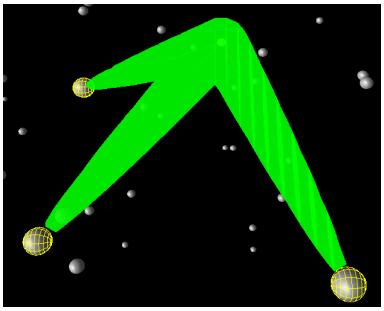


FIGURE 6.14 – Visuel d'un Starfish à trois branches, sans application du Ring Concept. La direction des branches est difficilement perceptible.

Afin de mieux distinguer les directions dans lesquelles les branches sont dirigées, nous avons choisi d'adapter le Ring Concept que nous avons décrit au chapitre 5. Chaque branche est découpée en un nombre fixé N_s de segments, ou *anneaux*. Un dégradé de couleur permet de distinguer les anneaux, comme illustré en figure 6.15. Les anneaux sont semi-transparents afin que l'utilisateur perçoive le noyau de Starfish à l'intérieur de la branche. La distinction entre les anneaux est renforcée en jouant sur leur transparence. Sur la figure 6.15, un anneau sur deux a une transparence plus élevée. Nous pensons que le dégradé de couleur aide l'utilisateur à distinguer les branches entre elles, tandis que la succession d'anneaux indique la direction pointée par une branche (voir le principe du Ring Concept, en section 5.1).

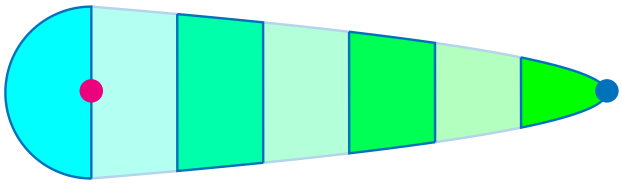


FIGURE 6.15 – Aperçu du visuel d'une branche de Starfish. La branche est découpée en plusieurs segments, avec un dégradé de couleurs. La transparence des segments permet de mieux distinguer les anneaux entre eux.

À chaque instant, nous reconstruisons la surface de Starfish. Nous commençons par appliquer un algorithme de Marching Cubes [Lorensen 87] sur le champ de potentiel généré par le squelette, pour produire un maillage triangulaire de la surface S_{ψ_0}. Nous

notons B_i la branche pour laquelle la cible C_i est l'extrémité. Nous disons qu'un point de l'espace appartient à la branche B_i s'il est plus proche du segment $[N_0C_i]$ que de tout autre segment $[N_0C_j]$, au sens de la distance euclidienne :

$$\forall X \in \mathbb{R}^3, X \in B \text{ si et seulement si } B = B_{\underset{j}{\operatorname{argmin}}\, d(X,[N_0C_j])} \tag{6.14}$$

Nous notons B_X la branche à laquelle appartient le point X, et i_X l'entier tel que $B_X = B_{i_X}$. De la même manière, chaque branche étant découpée en N_s anneaux, chacun des points appartenant à cette branche appartient à un anneau. Pour toute branche B_i et pour tout k tel que $0 \leqslant k < N_s$, nous définissons l'anneau $A_{i,k}$ de la branche B_i de la manière suivante :

$$A_{i,k} = \begin{cases} \left\{X \in \mathbb{R}^3, \tau_i\left(X\right) < 0\right\} & \text{si } k = 0 \\ \left\{X \in \mathbb{R}^3, \dfrac{k-1}{N_s} \leqslant \tau_i\left(X\right) < \dfrac{k}{N_s}\right\} & \text{sinon.} \end{cases} \tag{6.15}$$

Nous notons A_X l'anneau auquel appartient le point X, et k_X l'entier tel que $A_X = A_{i_X,k_X}$. La couleur de chaque vertex V du maillage est déterminée par l'anneau auquel il appartient A_V, comme décrit par l'algorithme 6.1. La couleur d'un anneau est une combinaison linéaire de deux couleurs c_0 et c_{N_s-1} fixées au préalable, représentant la couleur respectivement du premier et du dernier anneau. La figure 6.16 montre un Starfish possédant 3 branches, dans son environnement $3D$. Pour des raisons de confort visuel et de compréhension, nous conseillons d'utiliser deux paires de couleur (c_0, c_{N_s-1}) selon que Starfish est en mode ***Move*** ou ***Select***.

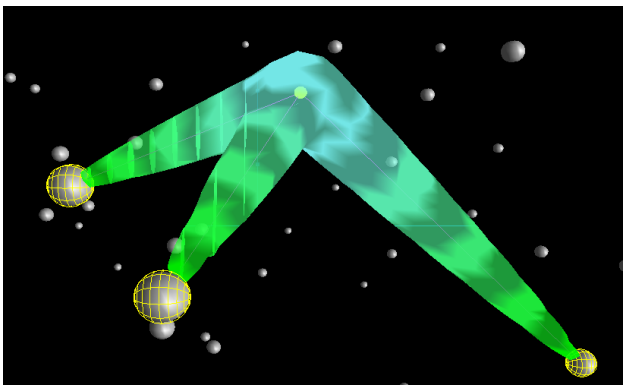


FIGURE 6.16 – Illustration $3D$ de Starfish, avec 3 branches.

Algorithme 6.1 Couleur d'un sommet de la surface de Starfish

Entrée : Un sommet (ou vertex) V de la surface S_{ψ_0}.
Sortie : La couleur c_V et la transparence α_V de ce sommet.

// Nous vérifions de quelle branche le sommet V est le plus proche,
// au sens de la distance euclidienne.
// Nous obtenons ainsi l'entier i_V (voir l'équation (6.14)).

// Nous déterminons la position de V dans sa branche.
// Nous obtenons ainsi le réel $\tau_{i_V}(V)$ (voir l'équation (6.8)).

// Nous déterminons à quel anneau A_V de la branche appartient V.
// Nous obtenons ainsi l'entier k_V (voir l'équation (6.15)).

// Tous les sommets du premier anneau (respectivement le dernier anneau) ont
// la même couleur c_0 (respectivement c_{N_s-1}) fixée au préalable.
// La couleur des autres anneaux dépend linéairement de ces deux couleurs.

$$c_V \longleftarrow \left(1 - \frac{k_V}{N_s - 1}\right) \cdot c_0 + \left(\frac{k_V}{N_s - 1}\right) \cdot c_{N_s-1}$$

// Chaque sommet du maillage est semi-transparent.
// Deux valeurs de transparence α_p et α_i sont prédéterminées afin de fixer
// la transparence respectivement des anneaux pairs et impairs.

$$\alpha_V \longleftarrow \begin{cases} \alpha_p & \text{si } k_V \text{ est pair,} \\ \alpha_i & \text{sinon.} \end{cases}$$

6.3.3 Contrainte du noyau

Dans le mode ***Select***, la surface de Starfish reste fixe et n'est plus modifiée par le déplacement du noyau. Le noyau est manipulé par l'utilisateur mais est contraint à l'intérieur du volume formée par la surface, afin qu'il puisse glisser sur la surface, et s'arrêter sur les cibles capturées. Cependant, la main de l'utilisateur contrôlant le périphérique d'interaction ne peut être contraint sans un dispositif haptique. Pour contourner cette limitation, nous avons développé un algorithme permettant de lier le pointeur $3D$, *libre*, manipulé par l'utilisateur à un second pointeur, dit *captif*, dont le but est de suivre le pointeur libre tout en restant contraint dans le volume V_{ψ_0} de Starfish. Lorsque l'utilisateur active le mode ***Select***, le pointeur captif est créé à la même position que le pointeur libre. Tant que le pointeur libre est à l'intérieur de V_{ψ_0}, leurs positions sont identiques. Lorsque le pointeur libre traverse la surface S_{ψ_0}, le pointeur captif doit glisser le long de la surface pour suivre le pointeur libre.

Une méthode simple pour obtenir ce comportement serait de placer le pointeur captif à chaque instant sur le point de la surface S_{ψ_0} le plus proche du pointeur libre. Cepen-

dant, cette méthode peut entraîner des sauts involontaires entre deux branches, comme l'illustre la figure 6.17. Le mouvement du pointeur captif est donc discontinu. Ce comportement peut perturber l'utilisateur et rendre le déplacement du pointeur laborieux, en particulier lorsque ces sauts sont induits par les tremblements naturels de la main. Nous pensons que le pointeur captif devrait glisser de manière continue pour éviter les sauts.

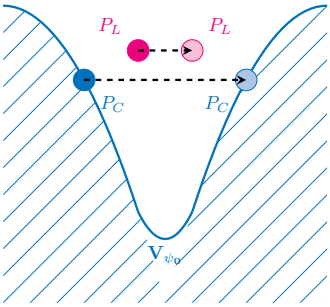


FIGURE 6.17 – Saut entre deux branches dans le cas où le pointeur captif (P_C) est placé sur le point de la surface le plus proche du pointeur libre (P_L). Le mouvement du pointeur captif n'est pas continu.

Nous proposons un algorithme pour suivre la surface, basée sur de petits mouvements successifs le long de la surface. Cette méthode (voir Algorithme 6.2) est illustrée en figure 6.18.

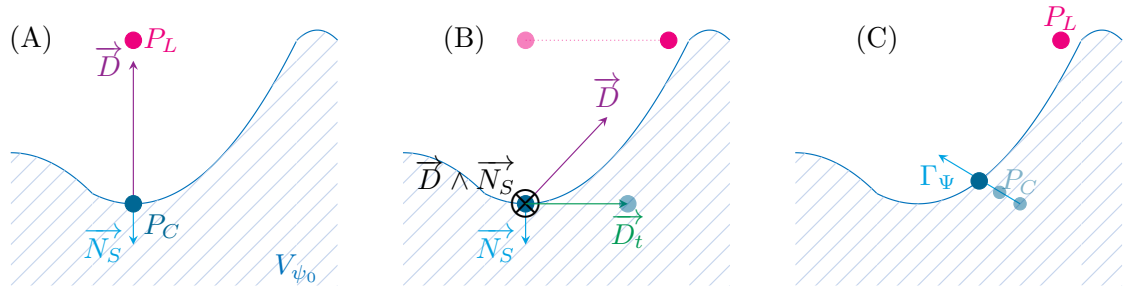


FIGURE 6.18 – Comportement du pointeur captif pour le suivi de surface. **(A) :** $\overrightarrow{D}$ et $\overrightarrow{N_S}$ sont colinéaires. Le pointeur captif n'est pas déplacé. **(B) :** L'utilisateur a déplacé le pointeur libre. $\overrightarrow{D}$ et $\overrightarrow{N_S}$ ne sont plus colinéaires. Le pointeur captif est translaté d'un vecteur $\overrightarrow{D_t}$. **(C) :** Le pointeur captif est itérativement déplacé selon le gradient du champ de potentiel, afin de revenir sur la surface.

Algorithme 6.2 Suivi de surface pour le pointeur captif

Initialisation :
$P_C \longleftarrow$ Position du pointeur captif
$P_L \longleftarrow$ Position du pointeur libre
$\overrightarrow{D} \longleftarrow \overrightarrow{P_C P_L}$
$\overrightarrow{N_S} \longleftarrow$ Vecteur normal à la surface en P_C

// Si le pointeur libre est à l'intérieur du volume,
// le pointeur captif est replacé sur le pointeur libre.
Si $\Psi(P_L) \geqslant \psi_0$
    $P_C \longleftarrow P_L$
    Arrêt de l'algorithme
Fin Si

// Condition d'arrêt :
Si $\overrightarrow{D}$ et $\overrightarrow{N_S}$ sont colinéaires (voir figure 6.18A)
    // Le pointeur captif n'est pas déplacé.
    Arrêt de l'algorithme
Fin Si

// Le pointeur captif est translaté d'un vecteur $\overrightarrow{D_t}$ *(voir figure 6.18B) :*
$\overrightarrow{D_t} \longleftarrow \delta_t \cdot \overrightarrow{N_S} \wedge \left(\overrightarrow{D} \wedge \overrightarrow{N_S}\right)$
$P_C \longleftarrow P_C + \overrightarrow{D_t}$

// Le pointeur captif est ensuite itérativement déplacé selon le gradient
// du champ de potentiel, pour revenir sur la surface (voir figure 6.18C).
Tant que $\Psi(P_C) \neq \psi_0$
    $\Gamma_\Psi \longleftarrow$ Gradient du champ de potentiel en P_C
    Si $\Psi(P_C) > \psi_0$
        $P_C \longleftarrow -\delta_g \Gamma_\Psi$
    Sinon
        $P_C \longleftarrow +\delta_g \Gamma_\Psi$
    Fin Si
Fin Tant que

// On recommence tant qu'une condition d'arrêt n'est pas atteinte.
Retour à Initialisation.

L'Algorithme 6.2 nécessite l'utilisation de deux paramètres, δ_t et δ_g. Le facteur de translation δ_t limite le déplacement du pointeur captif à chaque étape de l'algorithme, et définit donc sa vitesse. Deux considérations permettent de fixer ce paramètre. D'une part,

un facteur de translation trop petit implique de nombreuses boucles dans l'algorithme avant de vérifier la condition d'arrêt. Il en résulte pour l'utilisateur une sensation de lenteur de réactivité du pointeur captif. À l'inverse, un facteur de translation trop grand empêche le pointeur captif de se stabiliser sur une position d'arrêt. De plus, celui-ci peut alors être déplacé dans une zone dans laquelle le champ de potentiel est localement nul, avec un gradient nul (voir l'équation (6.16)), et restera bloqué à cette position. Le facteur d'ajustement δ_g, quant à lui, influe sur le nombre d'étapes nécessaires pour que le pointeur captif retourne sur la surface. Les mêmes considérations que pour le facteur de translation s'appliquent au facteur d'ajustement.

Le vecteur $\overrightarrow{N_S}$ est le vecteur normal de la surface en P_C. La surface étant définie comme l'isopotentielle de niveau ψ_0, ce vecteur est égal au gradient du champ de potentiel Γ_Ψ en P_C. Soit $\left(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k}\right)$ une base canonique de l'espace $\mathbb{R}^3$, et ϵ_g un facteur arbitrairement petit. Pour tout point de l'espace X, nous obtenons le gradient en X en calculant les potentiels des points autour de celui-ci :

$$\begin{aligned}\Gamma_{\Psi|X} = &\left(\Psi\left(X + \epsilon_g \overrightarrow{i}\right) - \Psi\left(X - \epsilon_g \overrightarrow{i}\right)\right)\overrightarrow{i} \\ &+ \left(\Psi\left(X + \epsilon_g \overrightarrow{j}\right) - \Psi\left(X - \epsilon_g \overrightarrow{j}\right)\right)\overrightarrow{j} \\ &+ \left(\Psi\left(X + \epsilon_g \overrightarrow{k}\right) - \Psi\left(X - \epsilon_g \overrightarrow{k}\right)\right)\overrightarrow{k}\end{aligned} \tag{6.16}$$

6.3.4 Sélection de cible

Un objet de la scène ne peut être sélectionné que si le mode ***Select*** est activé, c'est-à-dire lorsque le bouton **Select** est pressé. Lorsque l'utilisateur relâche ce bouton, Starfish vérifie si une cible est *atteinte* par l'utilisateur. Une cible C_i, à l'extrémité de la branche B_i, est atteinte si le pointeur captif a pénétré suffisamment profondément dans cette branche, *i.e.* les deux conditions suivantes sont respectées :

1. $B_i = B_{P_C}$ (voir l'équation (6.14))
2. $\tau_i(P_C) \geqslant \tau_{\min}$, où $\tau_{\min}$ est un facteur arbitraire entre 0 et 1.

Si une cible est atteinte, alors elle est sélectionnée, et l'utilisateur peut interagir avec elle, en fonction de l'application. Si aucune cible n'est atteinte, Starfish retourne en mode ***Idle***, tant que les boutons **Select** et **Move** ne sont pas pressés. Dans notre implémentation, nous avons fixé $\tau_{\min} = 0.3$, ce qui signifie que l'utilisateur peut sélectionner une cible si son pointeur captif a atteint 30% de la branche correspondante. Cette valeur permet également de définir une zone neutre de confort au centre de Starfish, permettant d'éviter de sélectionner involontairement une cible.

6.4 Évaluation de Starfish

Nous détaillons dans cette section le protocole expérimental que nous avons mis en place pour quantifier l'aide apportée par Starfish lors d'une tâche de sélection dans un environnement de réalité virtuelle, et la comparer avec le Depth Ray [Grossman 06] que nous avons décrit en section 4.4. Plusieurs études ont en effet montré que cette technique était l'une des plus efficaces dans ce contexte [Vanacken 09].

Durant cette expérience, nous présentons à l'utilisateur 3 scènes de densités différentes, comme illustré en figure 6.19. Chaque scène est constituée d'un nuage de points placé devant l'utilisateur. Les points sont représentés par des petites sphères grises. L'utilisateur doit sélectionner les cibles qui lui sont indiquées, à l'aide de Starfish ou de Depth Ray. Les deux scènes les moins denses nous semblent être des scènes réalistes, tandis que la plus dense est un cas extrême pour tester les limites des techniques considérées. Afin de valider le Ring Concept dans ce contexte, nous proposons deux versions de Starfish, ne différant que par leur aspect visuel. L'une de ces versions est affichée avec des anneaux, tel que nous l'avons décrit en 6.3.2. Nous notons cette version SFA. L'autre version a une couleur uniforme semi-transparente. Nous la notons SFU. Dans la suite, nous désignons également le Depth Ray par DR.

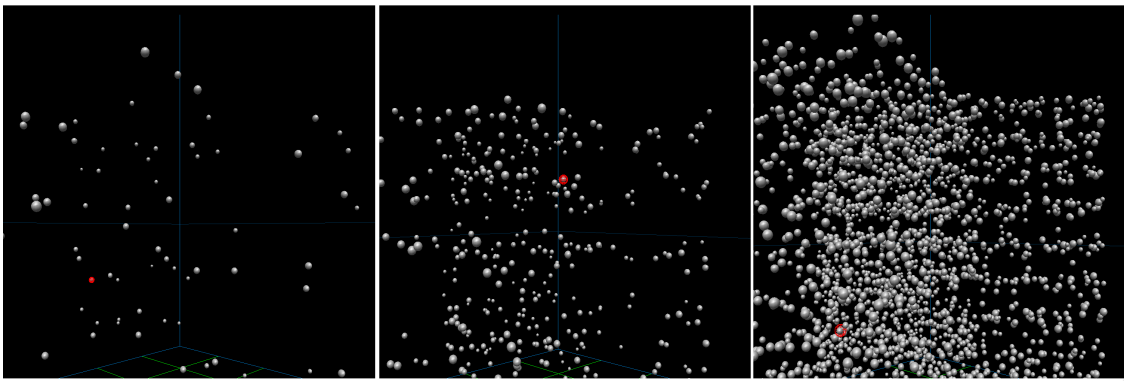


FIGURE 6.19 – Les trois nuages de points présentés à l'utilisateur. De gauche à droite : densité basse DB (75 cibles), moyenne DM (385 cibles) et élevée DE (2385 cibles).

6.4.1 Matériel et participants

Durant cette expérience, nous utilisons un Workbench semi-immersif, avec un affichage stéréoscopique actif de deux écrans (approximativement 2m de diagonale chacun), et un système de tracking repérant la position et l'orientation de l'utilisateur et du périphérique d'entrée. Le périphérique d'entrée est un flystick avec deux boutons. Seize (16) volontaires ont participé à cette expérience, 5 femmes et 11 hommes, entre 14 et 48

ans (pour une moyenne de 31 ans). Cinq (5) de ces participants (appelés dans la suite *débutants*) n'avaient jamais manipulé dans un environnement immersif, tandis que les onze (11) autres (*confirmés*) avaient réalisé plusieurs manipulations antérieurement.

6.4.2 Procédure

Il est demandé aux participants d'effectuer une tâche de sélection, divisée en 9 séries (3 techniques $\times$ 3 scènes) de 10 sélections chacune. Entre chaque série, une pause est autorisée afin de garder l'utilisateur concentré pendant toute la durée de la tâche. Un écran d'information détaillant les conditions de la prochaine série (technique et densité utilisées) est affiché pendant cette pause. Au début de chaque sélection, la cible à sélectionner est mise en évidence en changeant sa couleur en rouge, comme l'illustre la figure 6.19. Lorsque l'utilisateur sélectionne une des sphères, même s'il ne s'agit pas de la bonne cible, la sélection suivante commence immédiatement. Dix sélections successives constitue une série. À la fin des 9 séries, un questionnaire d'évaluation subjective est présenté aux participants. Durant cette expérience, nous nous intéressons au temps de sélection et au taux d'erreur pour les critères objectifs. L'ordre des séries est établi à l'aide d'un carré latin d'ordre 3, dont l'utilisation est rappelée en annexe A.

Au début d'une série, si la technique utilisée est Starfish (SFU ou SFA), celle-ci est placée au milieu du nuage de points. Entre deux sélections consécutives, la position de Starfish n'est pas modifiée. Le Depth Ray, quant à lui, est en permanence dirigé dans la direction désignée par le périphérique d'interaction. Sa longueur est fixée pour pouvoir traverser tout le nuage de points. Le marqueur de profondeur est placé au début du rayon (respectivement à l'extrémité du rayon) lorsque l'utilisateur place le flystick le long de son corps (respectivement lorsqu'il tend le bras). Entre ces deux cas extrêmes, la position du marqueur est linéaire en fonction de l'éloignement du flystick par rapport au corps. La longueur du rayon étant plus grande que la longueur du bras de l'utilisateur, la vitesse du marqueur est plus élevée que celle du flystick.

6.4.3 Conception

Les variables indépendantes de cette expérience sont :
- la technique de sélection T : Starfish classique SFA, Starfish uniforme SFU, et Depth Ray DR, et
- la densité de la scène D : basse DB, moyenne DM, et élevée DE.

Nous émettons les hypothèses suivantes sur les résultats de cette expérience :

H1 En basse densité, nous pensons que le Depth Ray permet de sélectionner plus rapidement, car il s'agit d'une technique de sélection directe, tandis que Starfish est une technique de raffinement progressif : une cible doit d'abord être présélectionnée avant de pouvoir être sélectionnée.

H2 En densité élevée, nous pensons que moins d'erreurs sont commises avec Starfish,

car cette technique n'est pas sensible aux tremblements de la main, et réduit les conséquences d'un geste de sélection peu précis.

H3 Nous prévoyons que Starfish procure un plus grand confort d'utilisation, car la technique est plus stable et permet une sélection avec peu d'efforts.

H4 En suivant le Ring Concept, nous estimons que la sélection avec SFA est plus rapide qu'avec SFU, car l'aide visuelle renforce l'information directionnelle fournie par les branches.

Chaque participant commence l'expérience par les trois séries d'une même technique, puis les trois séries d'une seconde technique, et enfin les trois séries de la troisième technique. L'ordre des techniques présentée à chaque utilisateur est établi par un carré latin d'ordre 3. Pour une technique donnée, l'ordre des trois séries (correspondant à trois densités différentes) est aléatoirement choisi, avec la condition que cet ordre n'ait pas été obtenu pour une technique précédente. Chaque participant débute l'expérience par une session d'entraînement, où chaque technique et chaque scène sont présentées à l'utilisateur. L'expérience complète dure approximativement 40 minutes.

6.5 Résultats

Une ANOVA a été réalisée sur les données récoltées pendant cette expérience, T et D étant les variables indépendantes, et le temps de sélection et le taux d'erreur les variables dépendantes.

Lorsque l'ANOVA révèle un impact significatif d'une variable indépendante, nous effectuons une analyse post-hoc par le test de Tukey HSD, afin d'identifier les différences significatives.

6.5.1 Temps de sélection

Le temps de sélection est défini comme le délai pour déplacer la technique et sélectionner la cible demandée. Les cas où l'utilisateur a sélectionné une mauvaise cible sont retirés de cette analyse. La table 6.1 regroupe les temps de sélection moyens en fonction des valeurs des deux variables indépendantes, tandis que la figure 6.20 présente les temps de sélection pour chaque série.

L'analyse statistique révèle que la densité de la scène D a un impact significatif ($F_{2,\,8} = 6.12$, $p < 0.05$) sur le temps de sélection. L'analyse de Tukey indique que le temps de sélection croît significativement en densité élevée, par rapport aux deux autres densités ($p < 0.05$). Toutefois, il ne semble pas y avoir de différences significatives entre la densité basse et la densité moyenne.

Aucun impact de la technique T sur le temps de sélection n'a été révélé par l'analyse statistique ($F_{2,\,8} = 2.59$, $p = 0.14$). Nous nous attendions à une sélection plus rapide du

Variable	Valeur	Temps (en secondes)
T	DR	4.20
	SFA	4.70
	SFU	4.90
D	DB	4.03
	DM	4.05
	DE	5.71
$T-D$	$DR-DB$	3.2
	$SFA-DB$	3.5
	$SFU-DB$	3.9
	$DR-DM$	3.2
	$SFA-DM$	3.6
	$SFU-DM$	3.8
	$DR-DE$	4.5
	$SFA-DE$	5.4
	$SFU-DE$	5.6

TABLE 6.1 – Temps de sélection moyens en fonction des variables indépendantes, obtenus lors de l'évaluation de Starfish

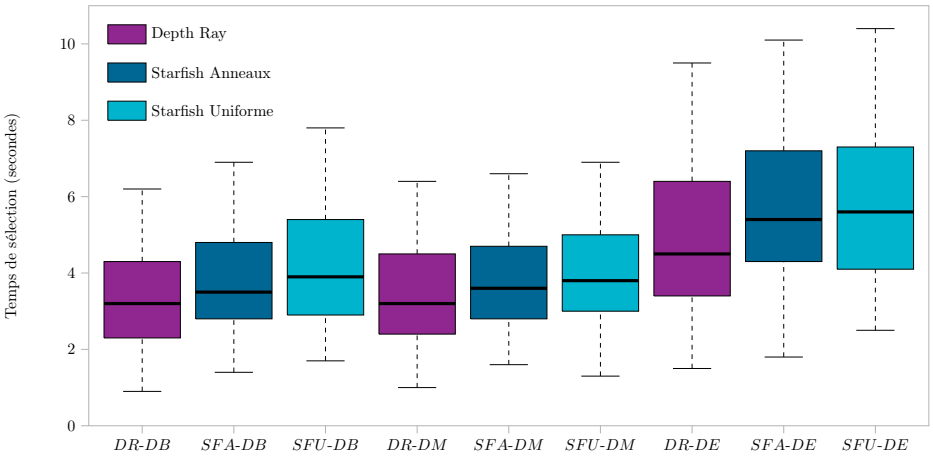


FIGURE 6.20 – Les temps de sélection pour chaque série $T-D$.

Depth Ray en densité basse, mais nous n'observons qu'une différence moyenne de 0.3s avec le Starfish classique, et de 0.7s avec le Starfish uniforme, différence qui n'est pas significative. Notre hypothèse **H1** est donc invalidée.

6.5.2 Taux d'erreurs

La table 6.2 regroupe les taux d'erreurs moyens en fonction des valeurs des deux variables indépendantes, tandis que la figure 6.21 présente visuellement les taux d'erreurs pour chaque série.

Variable	Valeur	Taux d'erreurs (en %)
T	DR	10.98
	SFA	2.45
	SFU	2.65
D	DB	3.43
	DM	2.94
	DE	9.71
$T-D$	$DR-DB$	3.53
	$SFA-DB$	3.53
	$SFU-DB$	3.24
	$DR-DM$	6.18
	$SFA-DM$	1.76
	$SFU-DM$	0.88
	$DR-DE$	23.24
	$SFA-DE$	2.65
	$SFU-DE$	3.24

TABLE 6.2 – Taux d'erreurs moyens en fonction des variables indépendantes, obtenus lors de l'évaluation de Starfish

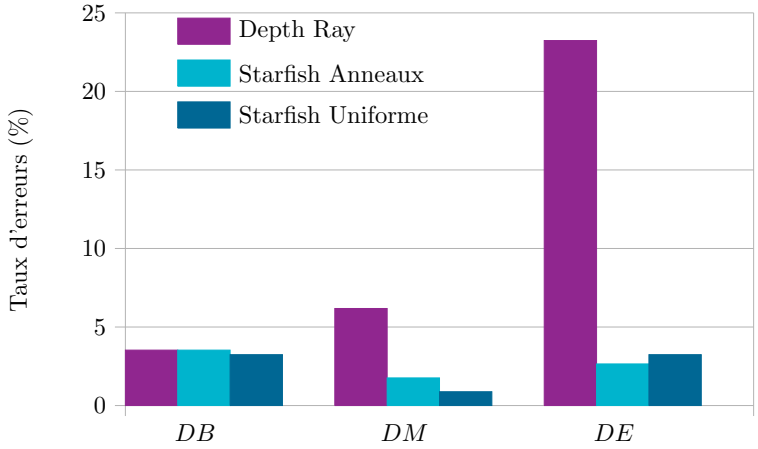


FIGURE 6.21 – Les taux d'erreurs pour chaque série $T-D$.

L'analyse statistique montre un impact significatif de la technique T ($F_{2,\ 8} = 52.4$, $p < 0.05$) et de la densité D ($F_{2,\ 8} = 31.45$, $p < 0.05$) sur le taux d'erreur.

L'analyse post-opératoire de comparaison des techniques révèle qu'en densité élevée, le taux d'erreur est significativement plus élevé pour Depth Ray que pour les deux versions de Starfish ($p < 0.05$), ce qui valide notre hypothèse **H2**. En densité moyenne, nous observons seulement une différence significative entre Depth Ray et le Starfish classique ($p < 0.05$). En densité basse, il n'y a aucune différence significative entre les techniques.

La comparaison des densités révèle seulement que pour le Depth Ray, le taux d'erreur croît significativement entre les deux densités les plus faibles, et la densité la plus élevée ($p < 0.05$). Nous avons en effet observé durant l'expérience que sélectionner une cible avec le Depth Ray dans la scène la plus dense était parfois si difficile que certains participants préféraient abandonner, et donc choisissaient volontairement de rater cette sélection. En revanche, il n'y a pas de différences significatives entre les densités, pour les deux versions de Starfish. Ce résultat montre que Starfish maintient un taux d'erreur stable, quelle que soit la densité de la scène.

6.5.3 Questionnaire subjectif

Un questionnaire est fourni aux participants à la fin du protocole. Nous leur demandons de répondre aux questions suivantes, présentées en utilisant une échelle de Likert à 7 points, où 1 est la réponse la plus négative, et 7 la plus positive :

Q1 Comment notez-vous l'intuitivité et le temps d'apprentissage de chaque technique ?

Q2 Comment estimez-vous votre performance pour chaque technique, en termes de rapidité, dans la scène à densité basse ?

Q3 Comment estimez-vous votre performance pour chaque technique, en termes de rapidité, dans la scène à densité haute ?

Q4 Globalement, comment notez-vous chaque technique ?

Les résultats de ce questionnaire sont détaillés en table 6.3. Nous présentons les scores moyens, ainsi que les moyennes séparées entre les participants débutants et confirmés. Ces résultats sont représentés visuellement en figure 6.22. Pour chaque question, le Starfish classique avec les anneaux a été en moyenne mieux noté que le Starfish uniforme, lui-même mieux noté que Depth Ray. Ce phénomène se retrouve également en observant seulement les participants débutants, ou seulement les participants confirmés.

6.6 Discussion

En marge du questionnaire, les participants sont invités à décrire leur ressenti pendant l'expérience. Ces remarques nous permettent d'interpréter les résultats obtenus. À

Question	Technique	Score Moyen	Score Débutants	Score Confirmés
Q1	DR	5.50	6.00	5.27
	SFU	6.31	6.80	6.09
	SFA	6.31	6.80	6.09
Q2	DR	4.00	6.40	2.91
	SFU	6.31	6.40	6.27
	SFA	6.38	6.60	6.27
Q3	DR	2.00	3.40	1.36
	SFU	5.75	5.40	5.91
	SFA	5.81	5.60	5.91
Q4	DR	2.19	4.00	1.36
	SFU	5.50	5.80	5.36
	SFA	6.06	6.80	5.73

TABLE 6.3 – Taux d'erreurs moyens en fonction des variables indépendantes, obtenus lors de l'évaluation de Starfish

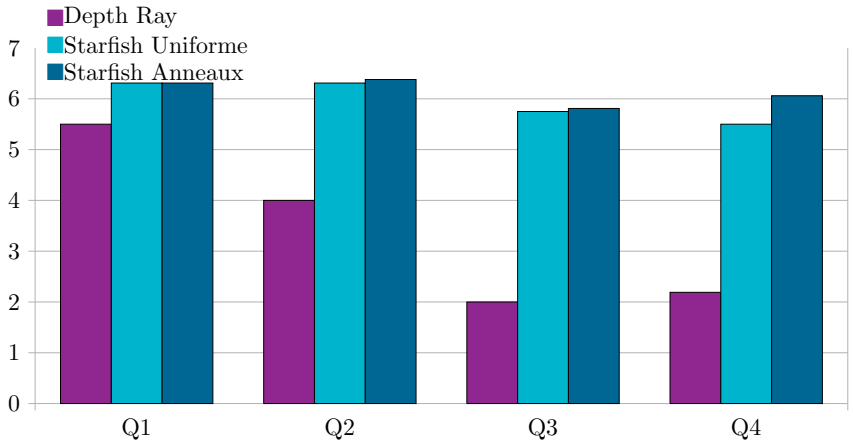


FIGURE 6.22 – Score moyen pour chaque technique pour le questionnaire présenté pendant l'évaluation de Starfish.

propos de la première question sur la prise en main de la technique, les participants soulignent que le Depth Ray est simple à comprendre, mais difficile à maîtriser, en particulier à cause des tremblements naturels de la main. Les utilisateurs confirmés remarquent également qu'en densité élevée, contrôler le marqueur de profondeur du Depth Ray revient à contrôler un simple pointeur $3D$ dans l'espace sans aucune aide. Au contraire, Starfish n'est pas sensible aux tremblements. Cette caractéristique est jugée très satisfaisante. De plus, plusieurs participants précisent que la forme de Starfish les aide à comprendre comment les cibles capturées sont placés. Aucune différence significative, en termes de temps

de sélection ou de taux d'erreur, n'est observée entre les deux versions de Starfish, ce qui invalide notre hypothèse **H4**. Cependant, les remarques post-expérimentales semblent indiquer que l'aide ressentie par la forme des branches est renforcée par les anneaux.

Nous remarquons un phénomène intéressant à propos du ressenti des participants. Bien que les résultats objectifs montrent que la sélection avec Depth Ray est en moyenne un peu plus rapide qu'avec Starfish (même si la différence n'est pas significative, voir figure 6.20), les participants ont eu la sensation d'être beaucoup plus lents avec Depth Ray (voir figure 6.22). Nous expliquons ce résultat de la manière suivante : avec la technique Depth Ray, l'utilisateur doit constamment être très précis dans ses gestes. Cette observation est vérifiée dans les scènes peu denses, mais est renforcée dans la scène très dense, car l'utilisateur est dans l'obligation de manipuler le marqueur de profondeur du rayon tout en gardant celui-ci dans la bonne direction. Cette difficulté conduit à une sensation de perte de temps, accentuée en haute densité où le taux d'erreur élevé génère de la frustration. Cependant, dans les faits, les temps de sélection entre Depth Ray et Starfish sont similaires, car cette dernière technique nécessite deux étapes avant de sélectionner une cible : une première étape où l'utilisateur s'approche de la cible, les filtres et le nombre de branches permettant de la présélectionner rapidement, et une seconde étape où le pointeur captif est déplacé pour pénétrer dans la bonne branche. Ces deux étapes s'effectue sans exiger une grande précision, et le taux d'erreur est moindre que pour Depth Ray. La sensation de perte de temps est donc moindre.

La dernière question est une conséquence des trois premières. En excluant un participant qui n'a pas de préférences entre les trois techniques, tous les autres donnent un meilleur score à Starfish qu'à Depth Ray. L'utilisation du Depth Ray est rapidement désagréable, tandis que le mode ***Idle*** de Starfish permet d'atténuer la fatigue physique. En conséquence, Starfish est considéré comme plus confortable à l'utilisation, ce qui valide notre hypothèse **H3**. Enfin, même si ce critère est difficile à quantifier, la plupart des participants déclarent que contrôler Starfish est « amusant » ou « *fun* ». En effet, le déplacement de cette technique dans un nuage de points évoque le déplacement d'une araignée dans sa toile. Or, l'utilisation de métaphores connues dans un système informatique, en particulier dans le domaine de la réalité virtuelle, permet d'accélérer la prise en main de la technique et d'améliorer les performances de l'utilisateur [Carroll 85, Stanney 95]. Ce phénomène explique les remarques des participants sur cette expérience.

6.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle technique d'interaction pour la sélection en environnement de réalité virtuelle. Cette technique, appelée Starfish, construit une surface implicite fermée autour du pointeur pour présélectionner les objets de la scène proches à l'aide de branches. Lorsque la cible qu'il souhaite atteindre est présélectionnée, l'utilisateur bloque la forme obtenue. Le pointeur est alors contraint à l'intérieur de la surface, et peut glisser sur celle-ci pour se diriger et s'arrêter sur la cible. Ce déplacement

ne nécessite pas d'être très précis, car la forme de branche dirige le pointeur directement sur la cible. Afin d'améliorer la perception de la scène, en particulier la disposition des cibles présélectionnées par rapport au pointeur, nous appliquons le Ring Concept (voir le chapitre 5) aux branches.

Nous avons mené une expérience afin de comparer les performances de Starfish avec le Depth Ray, une technique basée sur un rayon. Les résultats de cette expérience montrent que Starfish obtient des temps de sélection similaires à ceux du Depth Ray. Cependant, le taux d'erreur avec Starfish reste stable quelle que soit la densité de la scène, tandis que les performances de Depth Ray s'effondrent en densité élevée. De plus, Starfish est jugé plus confortable à contrôler. En effet, les guides permettent de sélectionner une cible sans effort et sans être gêné par les tremblements naturels de la main, tandis que les branches aident la planification du geste de sélection. Nous pensons donc que Starfish est une technique de sélection performante, et agréable à utiliser.

Une évolution naturelle pour Starfish serait l'ajout d'une composante haptique. En effet, la version que nous avons détaillée dans ce chapitre empêche le noyau de traverser la surface, mais cette contrainte est uniquement visuelle. Il en résulte des incohérences entre le mouvement de la main de l'utilisateur et celui du noyau. Cette incohérence ne semble en pratique pas perturbante pour l'utilisateur. Aucun des participants n'a en effet déploré de difficultés pour manipuler le noyau. Cependant, il est possible que le contrôle du noyau soit plus intuitif si la main de l'utilisateur est physiquement guidée et contrainte à l'intérieur de la surface implicite.

En terme de temps de sélection, notre expérience n'a révélé aucune différence significative entre le Depth Ray, une technique de rayon, et Starfish. Néanmoins, dans les conditions de notre expérience, l'utilisation du Depth Ray est en moyenne un peu plus rapide. Nous attribuons cette différence au fait que le Depth Ray est une technique de sélection directe, tandis que Starfish suit un processus de sélection en deux phases, une phase de présélection où les branches sont placées correctement pour capturer la cible à atteindre, puis une phase où le noyau est déplacé dans le volume. Nous pensons qu'un gain de temps est possible si nous pouvions prévoir quelle cible, parmi celles capturées, est celle à atteindre. Comme nous l'avons vu en section 4.2, les algorithmes de prédiction analysent le mouvement de l'utilisateur pour estimer la cible visée, ou son voisinage. Nous pourrions envisager d'appliquer cette approche au Starfish. Lorsque l'utilisateur bloque les branches, l'une d'elles, choisies par un algorithme de prédiction, pourrait par exemple être instantanément sélectionnable. Deux cas seraient alors possible. Si la branche choisie par l'algorithme correspond à la cible choisie par l'utilisateur, une simple validation (par l'appui sur un bouton) permet de sélectionner rapidement cette cible. Dans le cas contraire, l'utilisateur reprend le contrôle habituel de Starfish en déplaçant le noyau. Selon la qualité de l'algorithme de prédiction, nous pourrions alors observer une diminution conséquente du temps de sélection.

Néanmoins, les algorithmes de prédiction sont peu efficaces dans les espaces $3D$, de par la complexité des mouvements considérés. Dans les environnements $2D$, l'algorithme le plus performant actuellement est KEP [Ruiz 09]. Dans le chapitre suivant, nous proposons un nouvel algorithme de prédiction de cible, appelé SPEED. Cette méthode améliore les performances de KEP en découpant le geste de sélection $2D$ en une phase balistique et une phase de contrôle.

Chapitre 7

SPEED : Speed Profile sEparation for Endpoint Divination

I feel the need, the need for SPEED.

Pete « Maverick » Mitchell,
Top Gun

Sommaire

7.1 Profil de vitesse 126
7.2 Algorithme SPEED 127
7.2.1 Détection du pic 128
7.2.2 Distance de prédiction 129
7.2.3 Direction de prédiction 131
7.2.4 Position de la prédiction 132
7.2.5 Discussion sur les paramètres 132
7.3 Évaluation de SPEED 132
7.3.1 Matériel et participants 133
7.3.2 Procédure 133
7.4 Résultats et Discussion 134
7.5 Conclusion 136

Dans les chapitres 5 et 6, nous avons présenté deux nouvelles techniques d'interaction dans les environnements immersifs de réalité virtuelle, Bubble Bee et Starfish, exploitant toutes les deux notre méthode Ring Concept, basée sur des anneaux, afin de pointer une direction. Ces deux techniques de sélection sont construites autour d'un pointeur $3D$ que l'utilisateur manipule à l'aide d'un périphérique de type flystick. La vitesse de déplacement de ce pointeur dans l'espace est généralement fixée par un ratio constant avec la vitesse de déplacement du périphérique. Certaines méthodes modifie dynamiquement ce ratio en fonction de la position de la main comme le Go-Go [Poupyrev 96], ou

de sa vitesse comme PRISM [Frees 05]. Cette adaptation permet d'une part d'être plus précis lorsque le pointeur s'approche de la cible, et d'autre part d'augmenter l'espace de travail afin d'atteindre des cibles lointaines. Notre technique Starfish ne nécessite pas d'être précis pour sélectionner une cible proche, même dans une scène dense. La surface implicite contraint en effet le pointeur et le guide vers la cible désirée, corrigeant les mouvements de l'utilisateur. En revanche, si Starfish est efficace et offre un bon confort d'utilisation, nous observons une légère perte de performances en moyenne, en termes de temps de sélection, par rapport à Depth Ray, une technique de sélection basée rayon. Nous attribuons en partie cette différence au fait que Depth Ray permet de sélectionner en une seule phase, un clic sur la cible visée, tandis que Starfish nécessite deux phases, l'une pour placer correctement les branches, et l'autre pour déplacer le pointeur captif dans la branche souhaitée. Nous pensons qu'une solution à ce problème serait de proposer à l'utilisateur, lorsque celui-ci bloque la position des branches, de valider rapidement la sélection d'une cible bien choisie parmi celles capturées par les branches. Visuellement, la branche correspondant à cette cible pourrait être mise en surbrillance pour attirer l'attention de l'utilisateur. Deux cas peuvent alors se produire. Si nous choisissons la cible que l'utilisateur souhaite atteindre, il peut la sélectionner directement par un nouveau clic. Le déplacement du pointeur captif n'est donc plus nécessaire. Si en revanche le choix n'était pas le bon, l'utilisateur reprend le contrôle standard du Starfish, et déplace le pointeur captif jusqu'à la cible voulue.

Prédire la cible du mouvement de l'utilisateur, en analysant sa trajectoire et sa vitesse, nous semble être une solution pour choisir la cible à atteindre parmi celles capturées, et donc améliorer les performances de l'utilisateur. L'algorithme de Flash-Hogan [Flash 85], basé sur la loi du jerk minimal que nous avons évoquée en section 4.2, propose un modèle de trajectoire $3D$ rectiligne avec une courbe de vélocité en cloche. Néanmoins, si ce modèle est valide pour de petits mouvements, la trajectoire de la main est généralement courbée pour les mouvements plus amples auxquels nous nous intéressons. Dans ce contexte, une approche basée sur des courbes de Bézier semble plus pertinente [Faraway 07], mais reste peu performante pour prédire la cible d'arrivée.

La prédiction d'un geste de sélection 3D est un problème complexe dépendant de nombreux facteurs, comme la trajectoire non rectiligne du mouvement et l'absence de support physique. Pour cette raison, nous souhaitons réduire dans ce chapitre le nombre de ces facteurs, et nous limiter à des gestes de sélection dans des environnements $2D$, à l'aide d'une souris. Dans ce contexte, la trajectoire effectuée peut être considérée comme rectiligne, ce que supposent les méthodes de prédictions existantes. L'algorithme de prédiction existant le plus performant à notre connaissance, KEP [Ruiz 09], estime la distance de la cible en modélisant le profil de vitesse du mouvement par une courbe quadratique, et obtient sa position en considérant que ce mouvement est rectiligne. En retenant l'hypothèse qu'un tel mouvement se décompose en une phase d'approche rapide mais peu précise, et une phase de contrôle plus lente pour atteindre précisément la cible [Woodworth 99, Elliott 01], nous pensons qu'il est possible d'affiner ce modèle

et ainsi obtenir des prédictions plus précises. Nous décrivons dans ce chapitre ce nouvel algorithme de prédiction, SPEED, basé sur une détection du pic du profil de vitesse, et sur une modélisation de ce profil pendant la phase de contrôle du geste. Nous comparons ses performances en termes de taux de réussite avec celles de KEP, lors d'une tâche de sélection dans une scène $2D$.

7.1 Profil de vitesse

Une souris utilisée dans une application sur un poste de travail classique requiert une surface limitée pour être manipulé. À titre d'exemple, dans les conditions habituelles d'utilisation, l'empreinte de la souris sur un bureau lors d'une session de travail peut généralement s'inscrire dans un carré de 10cm de côté. Une étude plus détaillée des empreintes de différents périphériques d'interaction peut être trouvée dans [Card 90]. Dans ces conditions, nous considérons que la trajectoire d'un geste de sélection est rectiligne, comme le suppose également les algorithmes de prédiction existants [Asano 05, Lank 07, Ruiz 09]. Le profil de vitesse du pointeur de la souris est l'outil le plus analysé parmi les algorithmes de prédiction. En effet, la distance de la cible est fonction de l'amplitude du pic de vitesse [Walker 93]. L'algorithme Delphian Desktop [Asano 05] émet l'hypothèse qu'il s'agit d'une fonction affine dépendant de la direction et de l'utilisateur, et utilise cette caractéristique pour estimer la position de la cible. Le pointeur est alors « envoyé » à cette position. Cependant, cette méthode ne semble conduire à un gain de temps que pour des sélections de cibles lointaines. L'algorithme KEP exploite quant à lui l'intégralité du profil de vitesse [Ruiz 09], en modélisant la courbe de vitesse en fonction de la distance parcourue, par une courbe quadratique [Hogan 84]. À chaque instant, une nouvelle courbe quadratique est calculée. Une estimation de la cible est obtenue en considérant les racines de cette courbe. L'algorithme renvoie alors une prédiction lorsque cette estimation se stabilise.

Cet algorithme est à notre connaissance le plus performant en terme de taux de réussite. Cependant, nous pensons qu'il est possible d'obtenir de meilleurs résultats en découpant le geste en une phase balistique et une phase de contrôle [Woodworth 99]. Nous partons de l'hypothèse que l'utilisateur ne contrôle pas précisément sa vitesse lors de la phase balistique, mais seulement à partir de la phase de contrôle lorsque le pic de vitesse est atteint. Nous en concluons que le profil de vitesse n'est pas symétrique par rapport au pic de vitesse. Pour valider cette hypothèse, nous souhaitons recueillir et enregistrer des profils de vitesse de mouvements de sélection. Nous avons effectué une expérience préliminaire au cours de laquelle nous demandons aux participants de sélectionner dans une grille la case qui leur est indiquée. Cette expérience est similaire dans sa conception à celle que nous détaillerons en section 7.3. La figure 7.1 illustre l'une des courbes enregistrées. Nous observons, conformément à notre hypothèse, que le profil de vitesse n'est pas symétrique par rapport au pic de vitesse. Nous estimons donc que modéliser l'intégralité du profil de vitesse par une seule courbe quadratique pour prédire le point d'arrivée du moment, *i.e.* à quelle distance la vitesse devient nulle, est moins pertinent que modéliser

séparément la phase balistique (jusqu'au pic de vitesse) et la phase de contrôle (à partir du pic de vitesse) par deux courbes quadratiques distinctes, comme l'illustre la figure 7.2.

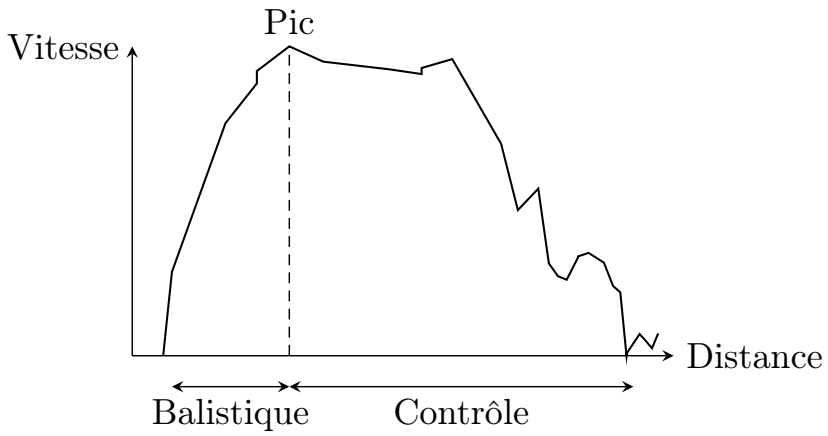


FIGURE 7.1 – Graphe d'un profil de vitesse du pointeur en fonction de la distance parcourue. Nous décomposons ce profil en deux phases, séparées par le pic de vitesse.

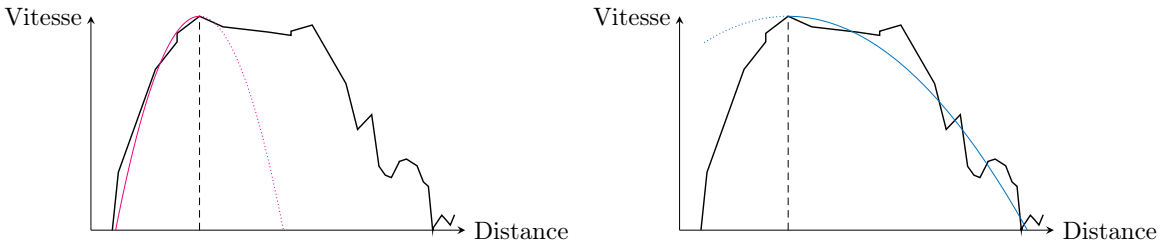


FIGURE 7.2 – Modélisation de la phase balistique (à gauche, en rouge) et de la phase de contrôle (à droite, en bleu) par deux courbes quadratiques.

7.2 Algorithme SPEED

Notre nouvel algorithme, que nous appelons *SPEED* pour *Speed Profile sEparation for Endpoint Divination*, est basé sur ce principe. À chaque instant après une détection de pic, nous approchons le profil de vitesse par une courbe quadratique. Nous obtenons alors la distance qu'il reste à parcourir avant la fin du mouvement. Ce procédé est sensible au bruit, et nécessite de nombreuses mesures pour obtenir un résultat satisfaisant et stable. Nous effectuons donc cette estimation de la distance à parcourir jusqu'à ce que

cette distance nous paraisse fiable. Nous en déduisons alors la position de la cible visée. Afin de distinguer deux gestes de sélection successifs, nous considérons qu'un geste commence lorsque la vitesse du pointeur dépasse un certain seuil V_0, et se termine lorsque la vitesse redescend en-dessous de ce même seuil. Chaque geste de sélection est traité indépendamment des autres. Lorsqu'un nouveau geste est détecté, l'algorithme suit les étapes suivantes :

1. **Détection du pic :** Nous cherchons le pic de vitesse afin de distinguer la phase balistique de la phase de contrôle.
2. **Distance de prédiction :** Lorsque le pic est détecté, nous prédisons la distance de la cible par une modélisation du profil de la phase de contrôle.
3. **Direction de prédiction :** Lorsque la distance de prédiction nous paraît fiable, nous analysons la trajectoire pour estimer dans quelle direction se trouve la cible.
4. **Position de la prédiction :** Nous exploitons enfin la distance et la direction estimées pour obtenir la position de la prédiction.

Ces étapes sont détaillées dans les parties suivantes.

7.2.1 Détection du pic

Dans la suite de ce document, nous notons (d_t, v_t) la donnée du profil de vitesse acquise à l'instant t [1], avec d_t la distance parcourue, et v_t la vitesse à cet instant. Nous notons également $(d_{\text{start}}, v_{\text{start}})$ la première donnée acquise du geste considéré (nous avons donc $v_{\text{start}} \geqslant V_0$), et P_{start} le point de la scène correspondant à cet instant. Nous appelons *pic local* une donnée $\left(d_{t_{\text{pic}}}, v_{t_{\text{pic}}}\right)$ du profil de vitesse telle que la vitesse en ce point soit supérieure aux vitesses acquises juste avant et juste après ce point, soit :

$$v_{t_{\text{pic}}} \geqslant \max\left(v_{t_{\text{pic}}+1}, v_{t_{\text{pic}}-1}\right) \tag{7.1}$$

Un pic local à l'instant t_{pic} ne peut donc être détecté qu'à l'instant $t_{\text{pic}} + 1$. Enfin, nous appelons *pic global* le pic local, parmi ceux détectés, dont la vitesse est la plus élevée. Nous le notons $\left(d_{t_{\text{pglob}}}, v_{t_{\text{pglob}}}\right)$. Nous notons également P_{pglob} le point de la scène correspondant au pic global. Ce pic global marque la séparation entre la phase balistique et la phase de contrôle, tant qu'un nouveau pic avec une vitesse plus élevée n'est pas détecté. La prédiction de la distance, décrite dans la partie suivante, exploite le profil de vitesse à partir de ce pic global. La figure 7.3 illustre cette détection de pic.

1. Pour simplifier la notation, les instants sont numérotés par des entiers. L'instant suivant l'instant t est donc l'instant $t + 1$.

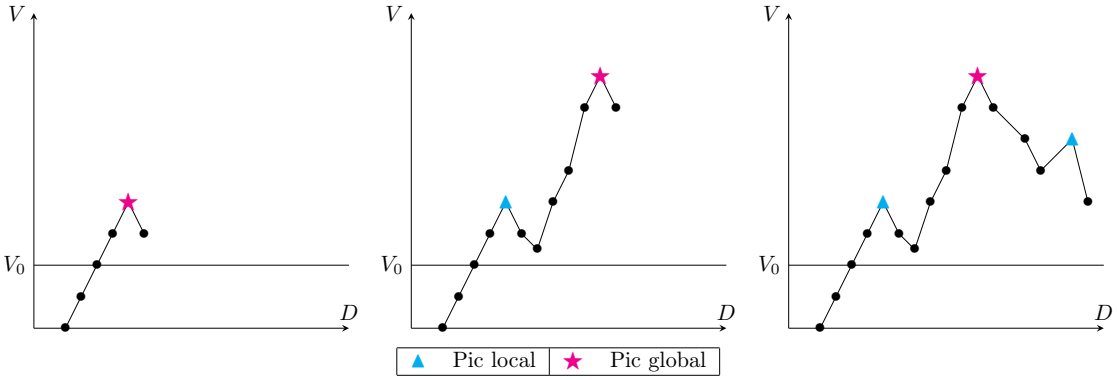


FIGURE 7.3 – Détection de pics. **Gauche :** Premier pic de vitesse détecté pour ce nouveau geste. Le pic est local et global. **Centre :** Nouveau pic local. Sa vitesse est supérieure à celle du pic global courant. Ce pic devient donc le nouveau pic global. **Droite :** Nouveau pic local. Sa vitesse est inférieure à celle du pic global courant, qui reste donc inchangé.

7.2.2 Distance de prédiction

À chaque instant t, si un pic global a déjà été détecté, nous considérons tous les points du profil de vitesse entre t_{pglob} et t. Nous approchons ces données par une courbe quadratique, à l'aide de la méthode des moindres carrés que nous rappelons en annexe B. Nous supposons que le profil de vitesse est quadratique à partir du pic global. Nous ajoutons donc la contrainte que la donnée $\left(d_{t_{\text{pglob}}}, v_{t_{\text{pglob}}}\right)$ soit le sommet de la parabole. Nous obtenons une équation de la forme :

$$v = \alpha d^2 + \beta d + \gamma. \tag{7.2}$$

Un pic n'étant détecté qu'à l'instant où la vitesse décroît, il est possible que cette équation n'admette pas de racines, comme illustré en figure 7.4. En effet, un pic global peut être détecté alors que la phase balistique n'est pas terminée. La vitesse croissant ensuite à nouveau, la méthode des moindres carrés peut alors retourner une équation du second degré sans racines. Dans ce cas, nous arrêtons le traitement et attendons la donnée suivante.

Dans le cas général, l'équation a deux racines, d_{r_1} et d_{r_2}, avec $d_{r_1} < d_{t_{\text{pglob}}} < d_{r_2}$. Cette seconde racine d_{r_2} correspond à la distance à laquelle nous estimons que la vitesse de l'utilisateur va s'annuler, donc à la fin de son geste. La distance $D_{\text{estimée}} = d_{r_2} - d_{\text{start}}$ est alors la distance estimée entre le début et la fin du geste. Le calcul de cette estimation est un procédé sensible au bruit. Avec trop peu de données, le calcul peut donner des résultats aberrants. À l'opposé, si nous attendons trop longtemps, nous perdons tout le

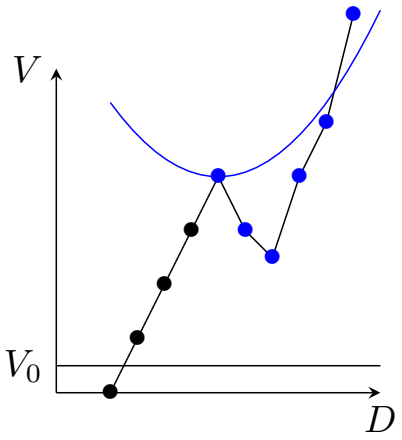


FIGURE 7.4 – Estimation d'une courbe quadratique non pertinente (en bleu). La phase balistique n'est pas encore achevée malgré la détection d'un pic local. Cette courbe ne permet pas d'obtenir une estimation de la distance de la cible.

bénéfice d'une prédiction. Nous considérons qu'une prédiction est fiable si le pointeur a déjà effectué un certain ratio de la distance prédite. Formellement, nous posons :

$$\rho = \frac{D_{\text{effectuée}}}{D_{\text{estimée}}} = \frac{d_t - d_{\text{start}}}{d_{r_2} - d_{\text{start}}}, \tag{7.3}$$

où $D_{\text{effectuée}} = d_t - d_{\text{start}}$ est la distance effectuée depuis le début du geste. Cette quantité ρ est la portion de distance réellement effectuée, par rapport à la distance totale estimée du geste. Une estimation est considérée comme fiable si $\rho \geqslant \rho_{\min}$, avec $\rho_{\min}$ une constante dont la valeur est fixée empiriquement. Nous appelons alors *distance de prédiction* la quantité $D_{\text{prédiction}} = d_{r_2} - d_{t_{\text{pglob}}}$, et nous notons P_0 le point de la scène pour lequel cette estimation fiable a été calculée. La figure 7.5 schématise les différentes quantités calculées pendant ce processus.

Note : La quantité ρ est un pourcentage dont la valeur est le plus souvent comprise entre 0 et 1. Néanmoins, nous pouvons obtenir $d_{r_2} < d_t$, comme l'illustre la figure 7.6, et donc $\rho > 1$. Nous interprétons simplement un tel phénomène par le fait que l'utilisateur a déjà parcouru une distance plus élevée que notre distance de prédiction. En d'autres termes, l'utilisateur a dépassé sa cible.

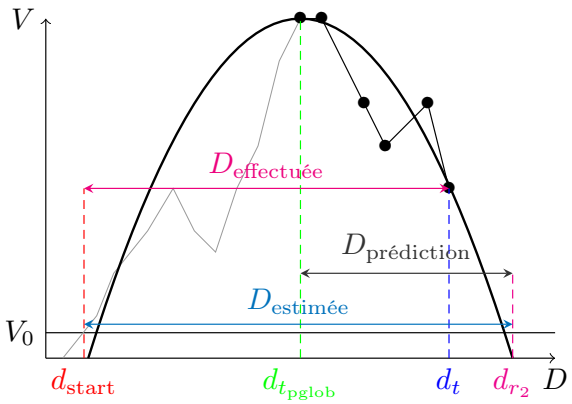


FIGURE 7.5 – Calcul de la distance de prédiction. Aperçu des différentes distances calculées. La distance $D_{\text{prédiction}}$ est utilisée lors de l'étape de prédiction de la cible.

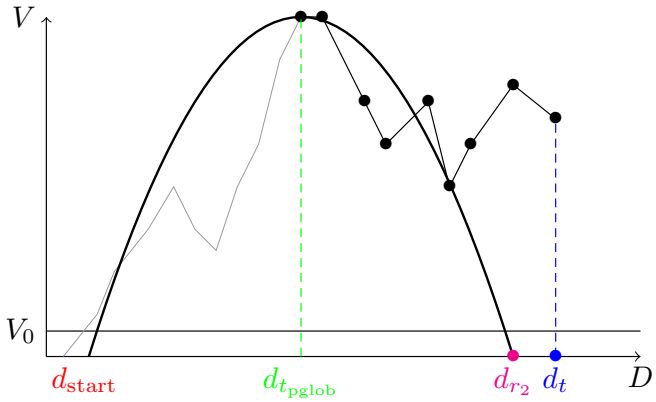


FIGURE 7.6 – Cas où l'utilisateur a dépassé sa cible. Nous avons $d_{r_2} < d_t$, et donc $\rho > 1$.

7.2.3 Direction de prédiction

Dans la partie précédente, nous avons restreint notre modèle quadratique à la phase de contrôle du profil de vitesse. De la même manière, nous faisons l'hypothèse que la trajectoire empruntée par le pointeur n'est rectiligne que durant cette phase, et non durant l'intégralité du geste de sélection. Nous évitons ainsi des résultats perturbés par un début de geste peu précis. Lorsqu'une distance de prédiction est obtenue, nous définissons la *direction de prédiction* comme étant le vecteur unitaire :

$$\overrightarrow{U_0} = \frac{\overrightarrow{P_{\text{pglob}}P_0}}{\|\overrightarrow{P_{\text{pglob}}P_0}\|} \tag{7.4}$$

7.2.4 Position de la prédiction

Enfin, la position dans le plan de notre point de prédiction $P_{\text{prédic}}$ est calculée par la relation :

$$P_{\text{prédic}} = P_{\text{pglob}} + D_{\text{prédiction}} \cdot \overrightarrow{U_0} \tag{7.5}$$

La figure 7.7 illustre la création de ce point.

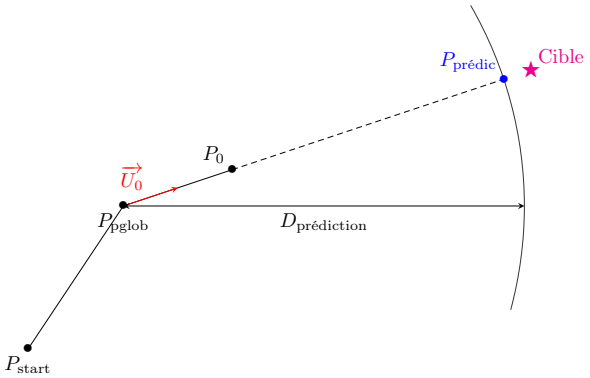


FIGURE 7.7 – Construction du point de prédiction, à partir du point de pic global, et à l'aide de la distance de prédiction et sa direction.

7.2.5 Discussion sur les paramètres

Notre algorithme SPEED dépend de deux facteurs, V_0 et ρ_{min}. D'une part, V_0 est le seuil de vitesse délimitant le début et la fin d'un geste. Or, plus une cible est lointaine, plus la vitesse maximale atteinte est élevée. À l'inverse, la vitesse maximale d'un mouvement de sélection pour une cible proche reste faible. Ainsi, une valeur trop élevée pour V_0 ne permet pas de détecter des gestes de sélection pour des cibles proches, car le mouvement n'atteint pas un tel seuil de vitesse. Lors de pré-tests, nous avons empiriquement fixé V_0 à 8mm/s. D'autre part, ρ_{min} représente le ratio de distance que l'utilisateur doit effectuer pour que la prédiction soit considérée comme fiable. Nous souhaitons comparer notre méthode avec l'algorithme actuellement le plus précis, KEP, qui prédit une cible à environ 85% du mouvement. Nous avons fixé ρ_{min} à 0.85. La section suivante détaille une expérimentation dans laquelle nous évaluons les performances de SPEED.

7.3 Évaluation de SPEED

Nous détaillons dans cette section le protocole expérimental que nous avons mis en place pour étudier la précision de notre algorithme de prédiction SPEED. Nous souhaitons vérifier si notre algorithme parvient à prédire la cible visée par l'utilisateur, ou, dans

le cas contraire, à quelle distance de la cible se situe la prédiction. Nous comparons ces résultats avec l'algorithme KEP [Ruiz 09]. Durant cette expérience, nous présentons à l'utilisateur une grille de cases dont deux sont mises en évidence, comme illustré en figure 7.8. Il est demandé à l'utilisateur de se placer sur l'une de ces cases, et d'effectuer un geste de sélection vers l'autre case. Nous n'effectuons aucun calcul durant l'expérience, mais recueillons simplement les données afin de leur appliquer les algorithmes SPEED et KEP en post-traitement. Ceci permet de comparer leur performance sur les mêmes jeux de données.

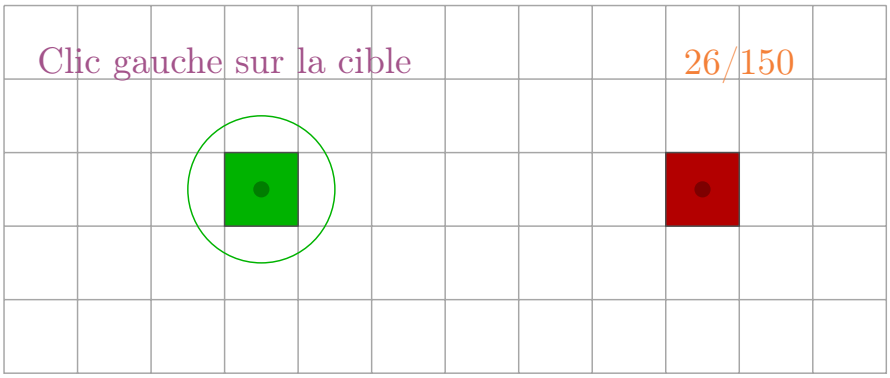


FIGURE 7.8 – Représentation de l'expérience pour l'évaluation de SPEED. L'utilisateur doit effectuer une alternance de clics entre deux cases d'une grille.

7.3.1 Matériel et participants

L'expérience a été réalisée sur une station de travail classique (Intel®Core™i5 CPU, 2.67GHz), avec un écran LCD 22" de résolution 1680 × 1050 (Samsung SyncMaster 223BW). Le périphérique d'entrée est une souris (Logitech RX250). Le ratio contrôle-affichage est constant et vaut 1 : 8. Les mouvements de la souris sont capturés à une fréquence moyenne de 100Hz. Dix (10) sujets ont participé à cette expérience. Tous sont droitiers et utilisent quotidiennement une souris.

7.3.2 Procédure

L'objectif de cette expérience est de quantifier la précision de notre algorithme. Nous nous plaçons dans les mêmes conditions que [Ruiz 09] pour comparer nos résultats. La tâche à effectuer est une tâche de sélection monodirectionnelle. Les sélections se font toujours dans la même direction droite-gauche. Les distances présentées entre la cible de départ et la cible d'arrivée sont 512, 1024 et 1536 pixels. Les cibles ont une taille de 16, 32, 64 et 128 pixels. Les $3 \times 4 = 12$ combinaisons Taille/Distance fournissent une gamme d'Indice de Difficulté (au sens de la loi de Fitts) entre 2.32 et 6.60. Lors de cette

expérience, nous avons fixé les constantes $V_0 = 8\text{mm/s}$ et $\rho_{\min} = 0.85$. L'expérience est composée d'une phase d'entraînement et une phase de test lors desquelles chaque combinaison Taille/Distance est présentée respectivement 2 et 10 fois à l'utilisateur. L'ordre des tailles de cible présentées aux utilisateurs est établi selon un carré latin de taille 4. Pour une taille de cible donnée, les $3 \times 10 = 30$ sélections sont présentées dans un ordre aléatoire. L'expérience complète dure environ 15 minutes.

L'environnement présenté à l'utilisateur est une grille de cases. L'une d'elles, en rouge, représente la case de départ. Une autre, en vert et sur la même ligne, représente la cible à sélectionner. Une sélection consiste en un clic droit d'initialisation sur la case de départ, puis un clic gauche classique de sélection sur la cible. L'alternance de clics droit et gauche permet d'éviter les double-clics involontaires que nous avons constaté lors de pré-tests. Les utilisateurs ne sont pas informés des calculs effectués en post-traitement par les algorithmes SPEED et KEP.

7.4 Résultats et Discussion

Une prédiction à l'intérieur de la cible visée est dite *Réussie*. Une prédiction dans une case adjacente à celle visée est dite *Proche*. La table 7.1 détaille le taux de prédictions réussies et proches pour chaque combinaison Taille/Distance, pour notre algorithme SPEED. Le taux de réussite varie de 26.2% pour l'indice de difficulté maximal 6.60, à 74.3% pour l'indice de difficulté minimal 2.32. La table 7.2 résume le taux de prédictions réussies et proches pour l'algorithme KEP, que nous avons développé pour le comparer à SPEED. La figure 7.9 illustre les différences du taux de réussite en fonction de l'indice de difficulté de la tâche.

Distance	512			1024			1536		
Taille	Réussi	Proche	**Somme**	Réussi	Proche	**Somme**	Réussi	Proche	**Somme**
16	32.1%	17.6%	**49.7**%	30.8%	14.2%	**45.0**%	26.2%	13.1%	**39.3**%
32	43.0%	19.4%	**62.4**%	38.1%	9.9%	**48.0**%	35.1%	15.2%	**50.3**%
64	50.5%	26.6%	**77.1**%	45.4%	17.3%	**62.7**%	46.5%	10.1%	**56.6**%
128	74.3%	22.3%	**96.6**%	60.9%	14.5%	**75.4**%	47.9%	14.2%	**72.1**%

TABLE 7.1 – Taux de réussite de SPEED pour chaque combinaison Taille/Distance.

Nous observons que hormis sur la tâche de difficulté minimale (cases de 128 pixels de côté à une distance de 512 pixels) où le taux de réussite est légèrement supérieur pour KEP, notre algorithme SPEED obtient toujours de meilleurs performances. Nous notons en particulier que dans la scène de difficulté maximale (cases de 16 pixels de côté à une distance de 1536 pixels), SPEED retourne une réponse correcte ou une case adjacente dans près de 40% des sélections. En distinguant les tailles de cibles, nous obtenons un gain moyen sur le taux de réussite de respectivement +27.3%, +31.4%, +27.1% et +17.3% sur

Distance	512			1024			1536		
Taille	Réussi	Proche	**Somme**	Réussi	Proche	**Somme**	Réussi	Proche	**Somme**
16	5.0%	30.2%	**35.2**%	2.0%	15.7%	**17.7**%	0.0%	9.1%	**9.1**%
32	17.0%	25.3%	**42.3**%	3.0%	20.2%	**23.2**%	2.0%	10.1%	**12.1**%
64	35.0%	34.6%	**69.6**%	16.0%	25.7%	**41.7**%	10.0%	12.1%	**22.1**%
128	76.0%	21.9%	**97.9**%	33.0%	42.0%	**75.0**%	22.0%	23.2%	**47.2**%

TABLE 7.2 – Taux de réussite de KEP pour chaque combinaison Taille/Distance.

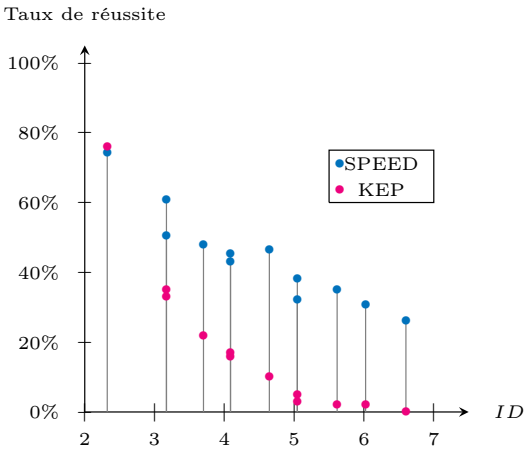


FIGURE 7.9 – Taux de réussite de SPEED et KEP en fonction des indices de difficulté.

les cibles de 16, 32, 64 et 128 pixels en considérant uniquement les prédictions réussies, et de +24.0%, +27.7%, +21.0% et +8.0% en considérant également les prédictions proches. Nous concluons de cette expérience que notre algorithme de prédiction est plus précis que KEP. Notre hypothèse de départ semble donc être confirmée ; durant un geste de sélection, l'utilisateur effectue un geste rapide peu précis, puis termine son mouvement en contrôlant sa vitesse. La séparation de ces deux phases permet à SPEED de modéliser plus justement le profil de vitesse.

Durant cette expérience, nous nous sommes contraints à une unique direction droite-gauche, afin de nous placer dans les mêmes conditions expérimentales que [Ruiz 09]. La direction du mouvement pourrait avoir une influence significative sur le taux de réussite de la prédiction. Ainsi, le Delphian Desktop [Asano 05] calibre son algorithme de prédiction en fonction des huit directions principales. De plus, il est montré que la sélection successive de plusieurs cibles [Huys 10] ou la présence de distracteurs [Blanch 11] perturbe le mouvement de l'utilisateur, invalidant la loi de Fitts. Conséquemment, ces paramètres peuvent également influer sur les performances des algorithmes de prédiction. Enfin, dans une application réelle, l'utilisateur est mentalement occupé à une autre

tâche. Nous pensons que ces conditions influent sur le contrôle de la vitesse, et donc sur les performances de SPEED.

Du passage à la $3D$

Ce document est principalement consacré à l'interaction en réalité virtuelle, et plus précisément à la tâche de sélection dans les environnements $3D$. Dans ce chapitre, nous avons présenté un algorithme de prédiction de cible dans le cadre de la sélection dans une application $2D$, pour un mouvement $1D$. Plusieurs facteurs limitent l'adaptation de SPEED dans une scène $3D$. En effet, le maniement de la souris et celui d'un flystick ne stimulent pas les mêmes groupes musculaires et les mêmes zones du squelette. Il en résulte que les mouvements de sélection $3D$ ne sont pas rectilignes, en particulier pour des mouvements en profondeur. De plus, les erreurs de perception dans ce type d'environnement faussent la planification du mouvement. Une phase de correction plus longue avec plusieurs modifications de trajectoires en découle [Liu 09a]. Nous en concluons que SPEED n'est pas adapté à ce type d'environnement.

7.5 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme de prédiction de cibles pour la sélection de cibles dans une scène $2D$. Cet algorithme, que nous appelons SPEED pour *Speed Profile sEparation for Endpoint Divination*, détecte le pic de vitesse du geste de sélection, séparant le profil de vitesse, c'est-à-dire la courbe de vitesse en fonction de la distance parcourue, en une phase balistique terminée par ce pic, et une phase de contrôle. SPEED modélise ensuite la phase de contrôle par une courbe quadratique. Ce modèle permet de déduire à quelle distance se situe la cible visée par l'utilisateur. L'algorithme considère que le mouvement est rectiligne, ce que nous pouvons supposer dans le cadre d'une interaction avec la souris, pour déterminer la position de la cible visée. Afin de réduire les imprécisions dues aux bruits dans les données recueillies, une prédiction n'est considérée comme fiable que si la distance réellement effectuée est suffisante par rapport à la distance prédite.

Nous avons mené une expérience afin de comparer les performances de SPEED avec KEP, un algorithme modélisant l'intégralité du profil de vitesse afin de déterminer la distance de la cible. Les résultats de cette expérience montrent que SPEED obtient un meilleur taux de réussite que KEP, pour plusieurs configurations de tailles de cible et de distances. En moyenne, nous observons un gain de $+25.8\%$ sur le taux de réussite en considérant uniquement les prédictions réussies, et de $+20.2\%$ en considérant également les prédictions proches. Ces performances nous permettent de valider notre modèle basé sur une séparation du profil de vitesse en une phase balistique et une phase de contrôle.

Les performances de notre algorithme n'ont été évaluées que dans le cas d'une scène

simple au cours d'une tâche de sélection monodirectionnelle. Nous souhaitons continuer son évaluation dans le cas de scènes plus complexes, et dans le cadre d'une application réelle, où l'utilisateur ne serait pas focalisé sur la seule tâche de sélection. En revanche, nous avons conduit des tests préliminaires sur l'exploitation de SPEED dans des environnements immersifs avec des périphériques d'interaction 3D, pour des résultats non satisfaisants. Nous expliquons les mauvaises performances de ces tests par la trajectoire non rectiligne des mouvements de sélection dans ces conditions, et par une mauvaise perception de la profondeur faussant la planification du geste.

Chapitre 8

Conclusion

I did it for me.
I liked it.
I was good at it.
And, I was really... I was alive.

Walter White,
Breaking Bad
Episode 5x16, *Felina*

Dans ce dernier chapitre, nous présentons un bilan de l'ensemble des travaux que nous avons effectués. Nous résumons les différentes techniques d'interaction que nous avons développées, ainsi que les résultats que nous avons obtenus lors de leur évaluation. Nous finirons en proposant quelques unes des perspectives que nous envisageons pour la suite de ce travail.

8.1 Bilan

Dans le cadre de ce travail, nous nous sommes intéressés à la tâche de sélection dans un environnement de réalité virtuelle, dans un contexte de visualisation et d'édition de maillages volumiques. Notre objectif était de proposer et d'évaluer des techniques d'interaction palliant les difficultés inhérentes à cette tâche, et ainsi d'améliorer les performances et le confort de l'utilisateur.

Dans un premier temps, nous avons décrit le concept de **réalité virtuelle** et listé les différentes catégories de **tâches** que pouvait effectuer un utilisateur dans un environnement immersif. La partie I était consacrée à cette vision globale du domaine qui nous intéresse.

Parmi ces tâches fondamentales de l'interaction, la **sélection** d'une entité de la scène virtuelle nous a particulièrement interpellé du fait des nombreuses problématiques qu'elle génère, et des obstacles qui s'y opposent. Nous avons donc dans un second temps, dans la

partie II, décrit ces différentes problématiques, comme la mauvaise perception de la profondeur dans ce type d'environnement ou les occlusions entre les objets de la scène. Afin de distinguer les différents types d'aides qu'il était possible de fournir à l'utilisateur pour améliorer ses performances, nous avons proposé un modèle, une **décomposition de la tâche de sélection** en trois phases. La **phase de décision** est une étape préliminaire au mouvement, où l'utilisateur évalue la disposition de la scène pour repérer sa cible. La **phase balistique** correspond à l'impulsion initiale du geste de sélection, celle dont l'amplitude de vitesse est la plus élevée. Enfin, l'utilisateur corrige sa trajectoire initiale durant la **phase de correction** afin d'atteindre précisément sa cible. Nous avons ainsi discerné trois axes d'aides possibles correspondant à ces trois phases. Nous avons ensuite décrit les différentes techniques existantes dans la littérature qui proposent de telles aides.

Le premier de ces axes est la **localisation** de la cible. En effet, bien percevoir la position de la cible dans la scène, relativement à la position de l'utilisateur et des autres objets de la scène, est une condition nécessaire pour décider d'une trajectoire correcte pour l'impulsion initiale du geste de sélection, avec une bonne amplitude de vitesse. De nombreuses techniques ont déjà été développées dans ce but, comme le radar affichant une vue aérienne 2D de la scène, ou la flèche indiquant la direction d'une cible.

Le second axe est la **prédiction**. La phase de correction étant la plus longue et la plus éprouvante pour l'utilisateur, l'analyse du mouvement pendant la phase balistique permet d'estimer la position de la cible visée ou son voisinage, et ainsi anticiper les corrections de mouvements. La plupart des algorithmes élaborés dans ce but exploite le profil de vitesse du pointeur pendant le mouvement, par exemple en liant linéairement l'amplitude du pic de vitesse avec la distance de la cible.

Enfin, le troisième axe concerne toutes les techniques améliorant les **performances** et le **confort** de l'utilisateur, en lui facilitant la phase de correction. Par exemple, les objets de la scène peuvent s'agrandir lorsque le pointeur se rapproche, réduisant la précision nécessaire à leur sélection.

Parmi les techniques existantes, nous avons identifié certaines limitations ou certains cas dans lesquels les aides proposées ne sont pas efficaces. Nous avons donc consacré la partie III aux solutions que nous proposons pour pallier ces limitations. Dans les trois chapitres de cette partie, nous avons détaillé le principe et l'implémentation des techniques que nous avons développées. Nous avons également décrit les protocoles expérimentaux que nous avons mis en place pour évaluer nos solutions et les comparer aux techniques existantes.

Dans le chapitre 5, nous avons observé que la multiplication des outils de l'utilisateur (comme une technique de sélection et une technique de localisation de cible) était un obstacle à l'interaction car elle favorise les occlusions dans la scène et diminue l'espace visuel. Nous avons donc introduit une nouvelle méthode pour augmenter un outil

quelconque manipulé par l'utilisateur, en lui ajoutant graphiquement une fonctionnalité supplémentaire, celle de désigner la direction dans laquelle se trouve une cible, comme le ferait par exemple une flèche 3D. Notre méthode, appelée **Ring Concept** est basée sur l'affichage d'anneaux alignés et orientés selon une même direction, définissant ainsi un axe. Un code couleur permet alors de distinguer les deux sens de cet axe. Afin d'évaluer cette solution, nous avons choisi d'augmenter le 3D Bubble Cursor, une technique de forme sphérique, pour former le **Bubble Bee**. Lors d'une expérience, nous avons évalué et comparé la capacité du Bubble Bee et d'une flèche 3D, technique la plus utilisée dans ce type de tâche, à pointer dans la direction d'une cible. Aucune différence significative n'a pu être trouvée entre les deux techniques. En effet, un utilisateur commet un nombre d'erreurs semblable quand il estime quelle est la cible pointée, et décide de cette cible dans un temps similaire. Nous en avons conclu que le Ring Concept est une bonne alternative à la flèche 3D, dans la mesure où leurs performances sont similaires. De plus, le Ring Concept a l'avantage de pouvoir être intégré à un autre outil, libérant ainsi l'espace visuel pour la scène.

Dans le chapitre 6, nous avons constaté qu'une interaction 3D libre était peu efficace pour une tâche de sélection. En effet, l'absence de support physique pour manipuler le périphérique d'interaction augmente la fatigue physique et les tremblements involontaires de la main. De fait, il est plus difficile pour l'utilisateur de sélectionner précisément sa cible. Nous en avons conclu que guider le pointeur en le contraignant dans une partie de l'espace bien choisie est une solution élégante et efficace pour améliorer le confort de l'utilisateur et réduire le temps de sélection. Nous avons alors développé la technique **Starfish**, dont le principe est le suivant. Starfish est basé sur la création d'une surface définie de manière **implicite**, fermée, centrée initialement sur le pointeur, et constituée de plusieurs **branches** se terminant sur des cibles proches. En déplaçant le pointeur, la surface est recrée dynamiquement pour être à nouveau centrée sur le pointeur. Ainsi, à chaque instant, plusieurs cibles sont atteintes par les branches. L'utilisateur peut bloquer la surface à tout moment ; le pointeur ne peut alors se déplacer dans l'espace qu'à l'intérieur du volume formé par cette surface, et peut donc glisser sur la surface des branches, pour atteindre l'une des cibles correspondantes. Il n'est donc pas nécessaire pour l'utilisateur d'être précis, son pointeur terminant son déplacement précisément au niveau de la cible en suivant la surface. Afin de mieux percevoir les directions empruntées par les branches, nous avons augmenté Starfish avec le Ring Concept. Nous avons comparé Starfish à une technique basée sur un rayon, dans une tâche de sélection dans un nuage de points. Nous avons constaté un temps de sélection similaire pour les deux techniques, tandis que le taux d'erreur de sélection est significativement plus élevée pour le rayon que pour Starfish, dans une scène très dense. De plus, un questionnaire subjectif et des remarques des participants de l'expérience nous ont permis de constater une nette préférence et un meilleur confort d'utilisation pour notre technique.

Dans le cas d'un geste de sélection pour une interaction et une scène en deux dimensions, l'algorithme de prédiction existant le plus performant, KEP, utilise un modèle

quadratique de l'impulsion initiale du profil de vitesse. Dans le chapitre 7, nous avons remarqué que cet algorithme peut être amélioré en ne modélisant par une courbe quadratique que la phase de contrôle de cette impulsion, la vitesse lors de la phase balistique étant moins contrôlée par l'utilisateur. Nous avons montré que notre algorithme, **SPEED**, obtient de meilleures performances que KEP en termes de taux d'erreurs. En effet, nous avons appliqué ces deux algorithmes sur les données recueillies lors d'un expérience demandant à l'utilisateur la sélection d'une case dans une grille, pour différentes tailles et distances de cases. Notre algorithme SPEED obtient presque systématiquement un meilleur taux de réussite. Ce résultat est valide si nous ne considérons que les prédictions exactes, ou si nous considérons également le voisinage de la cible. En d'autres termes, lorsque la prédiction échoue à estimer la bonne cible, SPEED estime un meilleur voisinage.

8.2 Perspectives

De nombreuses perspectives apparaissent à la suite du travail décrit dans ce document, notamment concernant l'amélioration des solutions que nous avons proposées.

8.2.1 Ring Concept

Nous avons validé le principe du **Ring Concept** au travers une expérience où nous l'avons appliqué sur une sphère pour que les anneaux pointent vers l'une des cibles. Lors du développement de Starfish, nous l'avons également appliqué à chacune des branches pour permettre une meilleure perception de leur orientation dans l'espace. Il serait maintenant intéressant de valider notre solution dans d'autres contextes que celui de la sélection, en augmentant d'autres outils de formes plus complexes avec le Ring Concept, par exemple une arme dans un jeu FPS (jeu de tir en vue subjective). À l'inverse, en conservant une forme sphérique, nous pensons que le Ring Concept en tant que widget d'orientation (comme nous l'avons décrit dans la conclusion du chapitre 5, voir la figure 5.10) est une solution efficace pour aligner plusieurs objets dans la même direction, ou faire pointer un objet vers un autre.

Nous avons proposé un code couleur simple, un dégradé de rouge et vert, pour distinguer les deux sens de l'axe défini par les anneaux. D'autres codes peuvent également être élaborés pour fournir d'autres indications, comme par exemple la distance de l'objet ciblé. Ainsi, l'une des couleurs extrêmes fixerait toujours la direction de la cible, tandis que l'autre indiquerait à quelle distance elle se trouve. Une étude serait nécessaire pour vérifier que l'ajout d'une nouvelle information supplémentaire n'entraîne pas une surcharge mentale trop élevée.

8.2.2 Starfish

Nous avons plusieurs perspectives concernant l'amélioration de **Starfish**, ou l'ajout de nouvelles fonctionnalités. L'extension la plus naturelle semble être l'ajout d'une com-

posante haptique, pour permettre à l'utilisateur de sentir la surface implicite, et ainsi guider sa main vers la cible (et non plus seulement son pointeur). Ceci permettrait d'éviter des incohérences entre le mouvement de la main et celui du pointeur.

Actuellement, la sélection d'un objet avec Starfish est un processus en deux étapes. L'utilisateur déplace dans un premier temps les branches jusqu'à ce que l'une d'elles atteigne la cible souhaitée, puis déplace son pointeur dans le volume obtenu. Nous pensons qu'il est possible d'améliorer les performances de l'utilisateur en pré-validant l'une des cibles capturées par les branches, en choisissant une bonne heuristique, basée par exemple sur la trajectoire du pointeur. Ainsi, dans la majorité des cas, il suffirait de confirmer la suggestion de cible, et donc éviter le déplacement du pointeur dans le volume.

Dans le cadre de notre travail, nous n'avons considéré que des cibles ponctuelles et statiques. Une perspective possible serait d'adapter Starfish pour sélectionner des cibles plus volumineuses ou surtout mobiles. Dans ce deuxième cas, nous pourrions garder un avatar des cibles capturées au niveau des branches, et lier visuellement ces avatars, par exemple par un segment, aux « vraies » cibles qui restent mobiles. Ainsi, il n'est pas nécessaire d'atteindre précisément sa cible, mais seulement de faire en sorte qu'elle soit capturée par l'une des branches lors de son mouvement, pour ensuite la sélectionner aisément avec le pointeur captif.

Dans un autre contexte applicatif, il serait intéressant d'évaluer une version 2D de Starfish, dans une application sur station de travail classique, avec un contrôle à la souris pour sélectionner une icône ou un item dans un menu circulaire. Il serait peut-être possible de sélectionner plus facilement dans le cas de scènes très denses.

8.2.3 SPEED

Si notre algorithme de prédiction **SPEED** obtient de très bons résultats par rapport aux algorithmes existants dans le cas d'une interaction 2D, nous pensons qu'il n'est pas adaptable à des scènes 3D. En effet, lors d'un test préliminaire, nous avons constaté qu'un modèle quadratique ne pouvait pas s'appliquer aux profils de vitesse de mouvements de sélection en trois dimensions. De plus, les trajectoires non rectilignes faussent également le calcul de la prédiction. Il est donc nécessaire d'élaborer un nouveau modèle du profil de vitesse plus adapté à ce type d'environnements.

Enfin, nous n'avons validé notre algorithme que dans une scène abstraite constituée d'une grille de cases. Il serait intéressant de le tester dans des applications réelles : outils de bureautique, jeux, ...

Bibliographie

[Accot 03] Johnny Accot & Shumin Zhai. *Refining Fitts' law models for bivariate pointing.* In Proceedings of the SIGCHI conference on Human factors in computing systems, 2003.

[Akkouche 01] Samir Akkouche & Eric Galin. *Adaptive implicit surface polygonization using marching triangles.* In Computer Graphics Forum, volume 20–2, pages 67–80. Wiley Online Library, 2001.

[Aretz 92] Anthony J Aretz & Christopher D Wickens. *The mental rotation of map displays.* Human Performance, vol. 5, no. 4, 1992.

[Argelaguet 09] Ferran Argelaguet & Carlos Andujar. *Efficient 3D pointing selection in cluttered virtual environments.* Computer Graphics and Applications, IEEE, vol. 29, no. 6, 2009.

[Arnaldi 06] Bruno Arnaldi, Philippe Fuchs & Pascal Guitton. Le Traité de la réalité virtuelle, Vol. Les applications de la réalité virtuelle, volume 4. Presses de l'Ecole des Mines, 2006.

[Arrouët 05] Cédric Arrouët, Marco Congedo, Jean-Eudes Marvie, Fabrice Lamarche, Anatole Lécuyer & Bruno Arnaldi. *Open-ViBE : A Three Dimensional Platform for Real-Time Neuroscience.* Journal of Neurotherapy, vol. 9, no. 1, 2005.

[Asano 05] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima & Fumio Kishino. *Predictive interaction using the delphian desktop.* In Proceedings of the 18th annual ACM symposium on User interface software and technology, UIST'05, 2005.

[Barr 81] Alan H Barr. *Superquadrics and angle-preserving transformations.* IEEE Computer graphics and Applications, vol. 1, no. 1, pages 11–23, 1981.

[Benko 07] Hrvoje Benko & Steven Feiner. *Balloon selection : A multi-finger technique for accurate low-fatigue 3d selection.* In IEEE Symposium on 3D User Interfaces 2007, 3DUI'07, 2007.

[Bharathan 13] Rasiah Bharathan, Saaliha Vali, Thomas Setchell, Tariq Miskry, Ara Darzi & Rajesh Aggarwal. *Psychomotor skills and cognitive load training on a virtual reality laparoscopic simulator for*

tubal surgery is effective. European Journal of Obstetrics & Gynecology and Reproductive Biology, 2013.

[Bideau 10] Benoit Bideau, Richard Kulpa, Nicolas Vignais, Sébastien Brault, Franck Multon & Cathy Craig. *Using virtual reality to analyze sports performance.* Computer Graphics and Applications, IEEE, vol. 30, no. 2, pages 14–21, 2010.

[Blanch 11] Renaud Blanch & Michael Ortega. *Benchmarking pointing techniques with distractors : adding a density factor to Fitts' pointing paradigm.* In Proceedings of the 2011 annual conference on Human factors in computing systems, 2011.

[Blinn 82] James F Blinn. *A generalization of algebraic surface drawing.* ACM Transactions on Graphics (TOG), vol. 1, no. 3, pages 235–256, 1982.

[Bouchard 06] Stéphane Bouchard, Sophie Côté, Julie St-Jacques, Geneviève Robillard & Patrice Renaud. *Effectiveness of virtual reality exposure in the treatment of arachnophobia using 3D games.* Technology and health care, vol. 14, no. 1, pages 19–27, 2006.

[Bourdot 10] Patrick Bourdot, Thomas Convard, Flavien Picon, Mehdi Ammi, D. Touraine & J. M. Vézien. *VR-CAD integration : Multimodal immersive interaction and advanced haptic paradigms for implicit edition of CAD models.* Comput. Aided Des., vol. 42, no. 5, 2010.

[Bowman 97] Doug A Bowman, David Koller & Larry F Hodges. *Travel in immersive virtual environments : An evaluation of viewpoint motion control techniques.* In Virtual Reality Annual International Symposium, pages 45–52. IEEE, 1997.

[Bowman 99a] Doug A Bowman, Donald B Johnson & Larry F Hodges. *Testbed evaluation of virtual environment interaction techniques.* In Proceedings of the ACM symposium on Virtual reality software and technology, VRST'99, pages 26–33. ACM, 1999.

[Bowman 99b] Douglas A. Bowman. *Interaction techniques for common tasks in immersive virtual environments : design, evaluation, and application.* PhD thesis, Georgia Institute of Technology, 1999.

[Bowman 01] Doug A. Bowman, Chadwick A. Wingrave, Joshua M. Campbell & Vinh Q. Ly. *Using Pinch Gloves for both Natural and Abstract Interaction Techniques in Virtual Environments.* In Proc. HCI International 2001, pages 629–633, 2001.

[Bowman 04] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola & Ivan Poupyrev. 3d user interfaces : Theory and practice. Addison Wesley Longman Publishing Co., Inc., 2004.

[Card 78] S. K. Card, W. K. English & B. J Burr. *Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT.* Ergonomics, vol. 21, no. 8, 1978.

[Card 90] Stuart K Card, Jock D Mackinlay & George G Robertson. *The design space of input devices.* In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 117–124. ACM, 1990.

[Carlin 97] Albert S. Carlin, Hunter G. Hoffman & Suzanne Weghorst. *Virtual reality and tactile augmentation in the treatment of spider phobia : a case report.* Behaviour Research and Therapy, vol. 35, no. 2, 1997.

[Carpendale 97] M Sheelagh T Carpendale, David J Cowperthwaite & F David Fracchia. *Extending distortion viewing from 2D to 3D.* Computer Graphics and Applications, IEEE, vol. 17, no. 4, 1997.

[Carroll 85] John M Carroll & Robert L Mack. *Metaphor, computing systems, and active learning.* International journal of man-machine studies, vol. 22, no. 1, pages 39–57, 1985.

[Cashion 12] Jeffrey Cashion, Chadwick Wingrave & JJ LaViola. *Dense and dynamic 3d selection for game-based virtual environments.* Visualization and Computer Graphics, IEEE Transactions on, vol. 18, no. 4, 2012.

[Chittaro 04] Luca Chittaro & Stefano Burigat. *3D location-pointing as a navigation aid in Virtual Environments.* In Proceedings of the working conference on Advanced visual interfaces, AVI'04, 2004.

[Coelho 09] Carlos M Coelho, Allison M Waters, Trevor J Hine & Guy Wallis. *The use of virtual reality in acrophobia research and treatment.* Journal of Anxiety disorders, vol. 23, no. 5, pages 563–574, 2009.

[Coffey 11] Dane Coffey, Nicholas Malbraaten, Trung Le, Iman Borazjani, Fotis Sotiropoulos & Daniel F Keefe. *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 2011, 2011.

[Convard 04] Thomas Convard & Patrick Bourdot. *History based reactive objects for immersive CAD.* In Proceedings of the 9th ACM symposium on Solid modeling and applications, SM '04, 2004.

[Covaci 13] Alexandra Covaci & Doru Talaba. *Towards Improvement in Free Throw Skills by the Means of Virtual Reality.* Global Journal on Technology, vol. 3, 2013.

[Cutting 97] James E. Cutting. *How the eye measures reality and virtual reality.* Behavior Research Methods, Instruments, & Computers, no. 1, 1997.

[Dang 07] Nguyen-Thong Dang. *A Survey and Classification of 3D Pointing Techniques.* In Proc. of IEEE Int. Conf. on Research, Innovation and Vision for the Future, 2007.

[Darken 97] Rudolph P Darken, William R Cockayne & David Carmein. *The omni-directional treadmill : a locomotion device for virtual worlds.* In Proceedings of the 10th annual ACM symposium on User interface software and technology, UIST'97, pages 213–221. ACM, 1997.

[Darken 99] Rudolph P Darken & Helsin Cevik. *Map usage in virtual environments : Orientation issues.* In Proceedings of IEEE Virtual Reality 1999, VR'99, 1999.

[De Haan 05] Gerwin De Haan, Michal Koutek & Frits H Post. *Intenselect : Using dynamic object rating for assisting 3d object selection.* In Proceedings of the 11th Eurographics conference on Virtual Environments, 2005.

[Elliott 01] Digby Elliott, Werner F Helsen & Romeo Chua. *A century later : Woodworth's (1899) two-component model of goal-directed aiming.* Psychological bulletin, vol. 127, no. 3, page 342, 2001.

[Elmqvist 05] Niklas Elmqvist. *BalloonProbe : Reducing occlusion in 3D using interactive space distortion.* In Proceedings of the ACM symposium on Virtual reality software and technology, 2005.

[Essert-Villard 09] Caroline Essert-Villard & Antonio Capobianco. *Hardborders : a new haptic approach for selection tasks in 3d menus.* In Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST'09, pages 243–244. ACM, 2009.

[Faraway 07] Julian J Faraway, Matthew P Reed & Jing Wang. *Modelling three-dimensional trajectories by using Bézier curves with application to hand motion.* Journal of the Royal Statistical Society : Series C (Applied Statistics), vol. 56, no. 5, pages 571–585, 2007.

[Ferley 00] Eric Ferley, Marie-Paule Cani & Jean-Dominique Gascuel. *Practical volumetric sculpting.* The Visual Computer, vol. 16, no. 8, pages 469–480, 2000.

[Fiorentino 03] Michele Fiorentino, Giuseppe Monno, Pietro A Renzulli & Antonio E Uva. *3d pointing in virtual reality : experimental study.* In XIII ADM-XV INGEGRAF International Conference on Tools And Methods Evolution In Engineering Design, 2003.

[FirsthandTechnology] FirsthandTechnology. *Cybertherapy for Phobias & PTSD.* Website, -. http ://www.firsthand.com/portfolio/cybertherapy.html.

[Fitts 54] Paul M. Fitts. *The information capacity of the human motor system in controlling the amplitude of movement.* Journal of Experimental Psychology, 1954.

[Flash 85] Tamar Flash & Neville Hogan. *The coordination of arm movements : an experimentally confirmed mathematical model.* The journal of Neuroscience, vol. 5, no. 7, pages 1688–1703, 1985.

[Frees 05] Scott Frees & G Drew Kessler. *Precise and rapid interaction through scaled manipulation in immersive virtual environments.* In Proceedings IEEE Virtual Reality 2005, VR'05, 2005.

[Fuchs 03] Philippe Fuchs, Bruno Arnaldi & Jacques Tisseau. Le Traité de la Réalité Virtuelle, volume 1. Presses de l'Ecole des Mines de Paris, 2003.

[Fuchs 06] Philippe Fuchs, Guillaume Moreau, Alain Berthoz & Vercher Jean-Louis. Le Traité de la Réalité Virtuelle. Mines Paris, 2006.

[Gaitatzes 01] Athanasios Gaitatzes, Dimitrios Christopoulos & Maria Roussou. *Reviving the past : cultural heritage meets virtual reality.* In Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage, VAST'01, pages 103–110. ACM, 2001.

[Galin 97] Éric Galin. *Métamorphose et visualisation de blobs à squelettes.* PhD thesis, Université Claude Bernard, Lyon, 1997.

[Galin 00] Eric Galin & Samir Akkouche. *Incremental polygonization of implicit surfaces.* Graphical Models, vol. 62, no. 1, pages 19–39, 2000.

[Garcia-Palacios 02] Azucena Garcia-Palacios, H Hoffman, A Carlin, TA Furness III & C Botella. *Virtual reality in the treatment of spider phobia : a controlled study.* Behaviour Research and Therapy, vol. 40, no. 9, 2002.

[Gélinas-Phaneuf 13] Nicholas Gélinas-Phaneuf, Nusrat Choudhury, Ahmed R Al-Habib, Anne Cabral, Etienne Nadeau, Vincent Mora, Valerie Pazos, Patricia Debergue, Robert DiRaddo & Rolando F Del Maestro. *Assessing performance in brain tumor resection using a novel virtual reality simulator.* International Journal of Computer Assisted Radiology and Surgery, pages 1–9, 2013.

[Geomagic] Geomagic. *Dispositifs tactiles Geomagic (anciennement Phantom SensAble).* Website, -. http ://geomagic.com/fr/products/phantom-omni/overview.

[Gerardi 10] Maryrose Gerardi, Judith Cukor, JoAnn Difede, Albert Rizzo & Barbara Olasov Rothbaum. *Virtual Reality Exposure Therapy for Post-Traumatic Stress Disorder and Other Anxiety Disorders.* Curr Psychiatry Rep, vol. 12, pages 298–305, 2010.

[Gerber 04] Dominique Gerber & Dominique Bechmann. *Design and evaluation of the ring menu in virtual environments.* Immersive projection technologies, 2004.

[Gerber 05] Dominique Gerber & Dominique Bechmann. *The spin menu : a menu system for virtual environments.* In Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality, pages 271–272. IEEE Computer Society, 2005.

[Gordon 94] James Gordon, Maria Felice Ghilardi, Scott E Cooper & Claude Ghez. *Accuracy of planar reaching movements.* Experimental Brain Research, vol. 99, no. 1, pages 112–130, 1994.

[Grosjean 01] Jérôme Grosjean & Sabine Coquillart. *Command & control cube : a shortcut paradigm for virtual environments.* In Proceedings of the 7th Eurographics conference on Virtual Environments & 5th Immersive Projection Technology, pages 1–12. Eurographics Association, 2001.

[Grosjean 03] Jérôme Grosjean. *Environnements virtuels : contrôle d'application et exploration de scènes 3D.* PhD thesis, Université de Versailles Saint Quentin en Yvelines, 2003.

[Grossman 04] Tovi Grossman & Ravin Balakrishnan. *Pointing at trivariate targets in 3D environments.* In Proceedings of the SIGCHI conference on Human factors in computing systems, 2004.

[Grossman 05] Tovi Grossman & Ravin Balakrishnan. *The Bubble Cursor : Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area.* In Proc. of SIGCHI conf. on Human factors in computing systems, CHI'05, 2005.

[Grossman 06] Tovi Grossman & Ravin Balakrishnan. *The design and evaluation of selection techniques for 3D volumetric displays.* In Proceedings of the 19th annual ACM symposium on User interface software and technology, UIST(06, 2006.

[Guiard 00] Cédric Guiard. *Modèles de surfaces déformables définis et contraints par un ensemble d'informations structurelles.* PhD thesis, Université Aix-Marseille 2, 2000.

[Gustafson 08] Sean Gustafson, Patrick Baudisch, Carl Gutwin & Pourang Irani. *Wedge : clutter-free visualization of off-screen locations.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'08, 2008.

[Haption] Haption. Website, -. http ://www.haption.com/site/index.php/fr/.

[Hasegawa 06] Shoichi Hasegawa, Ishikawa Toshiaki, Naoki Hashimoto, Marc Salvati, Hironori Mitake, Yasuharu Koike & Makoto Sato. *Human-scale haptic interaction with a reactive virtual human in a real-time physics simulator.* Computers in Entertainment (CIE), vol. 4, no. 3, 2006.

[Hérisson 04] Joan Hérisson, Pierre-Emmanuel Gros, Nicolas Férey, Olivier Magneau & Rachid Gherbi. *DNA in Virtuo : Visualization and*

exploration of 3D genomic structures. In Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, AFRIGRAPH '04, 2004.

[Hilton 96] Adrian Hilton, Andrew J Stoddart, John Illingworth & Terry Windeatt. *Marching triangles : range image fusion for complex object modelling.* In Proceedings of International Conference on Image Processing, volume 1, pages 381–384. IEEE, 1996.

[Hinckley 94] Ken Hinckley, Randy Pausch, John C Goble & Neal F Kassell. *A survey of design issues in spatial input.* In Proceedings of the 7th annual ACM symposium on User interface software and technology, 1994.

[Hogan 84] Neville Hogan. *An organizing principle for a class of voluntary movements.* The Journal of Neuroscience, vol. 4, no. 11, 1984.

[Hoppe 92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald & Werner Stuetzle. Surface reconstruction from unorganized points, volume 26. ACM, 1992.

[Huegel 09] Joel C Huegel, Andrew J Lynch & Marcia K O'Malley. *Validation of a smooth movement model for a human reaching task.* In IEEE International Conference on Rehabilitation Robotics, ICORR 2009, 2009.

[Huem Kwon 13] Joung Huem Kwon, John Powell & Alan Chalmers. *How level of realism influences anxiety in virtual reality environments for a job interview.* International Journal of Human-Computer Studies, 2013.

[Huys 10] Raoul Huys, Laure Fernandez, Reinoud J Bootsma & Viktor K Jirsa. *Fitts' law is not continuous in reciprocal aiming.* Proceedings of the Royal Society B : Biological Sciences, vol. 277, no. 1685, 2010.

[ICube] ICube. *Laboratoire ICube.* Website, -. http ://icube.unistra.fr/.

[Interrante 06] Victoria Interrante, Brian Ries & Lee Anderson. *Distance perception in immersive virtual environments, revisited.* In Virtual Reality Conference, 2006, VR'06, 2006.

[Iwase 01] H Iwase & A Murata. *Modelling of human's three-dimensional movement-Extending Fitts' model to three-dimensional pointing task.* In Proceedings. 10th IEEE International Workshop on Robot and Human Interactive Communication, 2001, 2001.

[Jo 11] Hyungeun Jo, Sungjae Hwang, Hyunwoo Park & Jung-hee Ryu. *Aroundplot : focus+ context interface for off-screen objects in 3D environments.* Computers & Graphics, vol. 35, no. 4, 2011.

[Kim 11] Joohwan Kim, Takashi Shibata, David Hoffman & Martin Banks. *Assessing vergence-accommodation conflict as a source*

of discomfort in stereo displays. Journal of Vision, vol. 11, no. 11, 2011.

[Kiyokawa 96] Kiyoshi Kiyokawa, Haruo Takemura, Yoshiaki Katayama, Hidehiko Iwasa & Naokazu Yokoya. *Vlego : A simple two-handed modeling environment based on toy blocks.* In Proc. of ACM Simpo. on Virtual Reality Software and Technology, VRST'96, pages 27–34. Citeseer, 1996.

[Klimmt 09] Christoph Klimmt, Christopher Blake, Dorothée Hefner, Peter Vorderer & Christian Roth. *Player Performance, Satisfaction, and Video Game Enjoyment.* Entertainment Computing–ICEC 2009, pages 1–12, 2009.

[Kopper 10] Regis Kopper, Doug A Bowman, Mara G Silva & Ryan P McMahan. *A human motor behavior model for distal pointing tasks.* International journal of human-computer studies, vol. 68, no. 10, 2010.

[Kopper 11] Regis Kopper, Felipe Bacim & Doug A Bowman. *Rapid and accurate 3D selection by progressive refinement.* In IEEE Symposium on 3D User Interfaces, 2011, 3DUI'2011, 2011.

[Krueger 91] Myron W Krueger. Artificial reality ii, volume 10. Addison-Wesley Reading (Ma), 1991.

[Kulik 09] A. Kulik, J. Hochstrate, A. Kunert & B. Froehlich. *The influence of input device characteristics on spatial perception in desktop-based 3D applications.* In IEEE Symposium on 3D User Interfaces, 2009., 3DUI 2009, 2009.

[Lank 07] Edward Lank, Yi-Chun Nikko Cheng & Jaime Ruiz. *Endpoint prediction using motion kinematics.* In Proceedings of the SIGCHI conference on Human factors in computing systems, 2007.

[Lécuyer 03] Anatole Lécuyer, Pascal Mobuchon, Christine Mégard, Jérôme Perret, Claude Andriot & J-P Colinot. *HOMERE : a multimodal system for visually impaired people to explore virtual environments.* In Virtual Reality, 2003. Proceedings. IEEE, pages 251–258. IEEE, 2003.

[Legendre 05] Adrien Marie Legendre. Nouvelles méthodes pour la détermination des orbites des cometes. F. Didot, 1805.

[Liang 94] Jiandong Liang & Mark Green. *JDCAD : A Highly Interactive 3D Modeling System.* Computer and Graphics, vol. 18, no. 4, pages 499–506, 1994.

[Liu 09a] Lei Liu, Robert van Liere, Catharina Nieuwenhuizen & J-B Martens. *Comparing aimed movements in the real world and in virtual reality.* In Virtual Reality Conference, pages 219–222. IEEE, 2009.

[Liu 09b] Yunzhou Liu, Fei Wang & Chensheng Wang. *The research on visual fatigue factor in stereoscopic image observation.* In IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design, 2009, CAID & CD 2009, 2009.

[Lorensen 87] William E Lorensen & Harvey E Cline. *Marching cubes : A high resolution 3D surface construction algorithm.* In ACM Siggraph Computer Graphics, volume 21–4, pages 163–169. ACM, 1987.

[Lourdeaux 01] Domitile Lourdeaux. *Réalité virtuelle et formation : conception d'environnements virtuels pédagogiques.* PhD thesis, École Nationale Supérieure des Mines de Paris, 2001.

[Lugrin 13] Jean-Luc Lugrin, Marc Cavazza, Fred Charles, Marc Le Renard, Jonathan Freeman & Jane Lessiter. *Immersive FPS games : user experience and performance.* In Proceedings of the 2013 ACM international workshop on Immersive media experiences, pages 7–12. ACM, 2013.

[MacKenzie 92a] I. S. MacKenzie & W. Buxton. *Extending Fitts' law to two-dimensional tasks.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92, 1992.

[MacKenzie 92b] I. Scott MacKenzie. *Fitts' law as a research and design tool in human-computer interaction.* Hum.-Comput. Interact., vol. 7, no. 1, 1992.

[Martinet 10] Anthony Martinet, Gery Casiez & Laurent Grisoni. *The design and evaluation of 3d positioning techniques for multi-touch displays.* In 3D User Interfaces, 3DUI'10, pages 115–118. IEEE, 2010.

[McCrae 10] James McCrae, Michael Glueck, Tovi Grossman, Azam Khan & Karan Singh. *Exploring the design space of multiscale 3D orientation.* In Proceedings of the International Conference on Advanced Visual Interfaces, AVI'10, 2010.

[McGuffin 05] Michael J McGuffin & Ravin Balakrishnan. *Fitts' law and expanding targets : Experimental studies and designs for user interfaces.* ACM Transactions on Computer-Human Interaction (TOCHI), vol. 12, no. 4, 2005.

[McMahan 12] Ryan P McMahan, Doug A Bowman, David J Zielinski & Rachael B Brady. *Evaluating display fidelity and interaction fidelity in a virtual reality game.* Visualization and Computer Graphics, IEEE Transactions on, vol. 18, no. 4, pages 626–633, 2012.

[Meyerbroeker 13] K Meyerbroeker, N Morina, GA Kerkhof & PMG Emmelkamp. *Virtual Reality Exposure Therapy Does Not Provide Any Additional Value in Agoraphobic Patients : A Randomized Controlled Trial.* Psychotherapy and psychosomatics, vol. 82, no. 3, pages 170–176, 2013.

[Milgram 94] Paul Milgram & Fumio Kishino. *A taxonomy of mixed reality visual displays.* IEICE TRANSACTIONS on Information and Systems, vol. 77, no. 12, 1994.

[Miller 05] Samuel A Miller, Noah J Misch & Aaron J Dalton. *Low-cost, portable, multi-wall virtual reality.* In Proceedings of the 11th Eurographics conference on Virtual Environments, pages 9–14. Eurographics Association, 2005.

[Mine 95] Mark R. Mine. *Virtual Environment Interaction Techniques.* Rapport technique, University of North Carolina, 1995.

[Monk 85] Donald L Monk. *Fitts' law in two dimensions with hand and head movements.* Journal of Motor Behavior, vol. 17, no. 1, 1985.

[Murata 98] Atsuo Murata. *Improvement of pointing time by predicting targets in pointing with a PC mouse.* International Journal of Human-Computer Interaction, vol. 10, no. 1, 1998.

[Murata 99] A. Murata. *Extending effective target width in Fitts' law to a two-dimensional pointing task.* International journal of human-computer interaction, vol. 11, no. 2, 1999.

[Nagel 08] Henrik R Nagel, Erik Granum, Søren Bovbjerg & Michael Vittrup. *Immersive visual data mining : the 3DVDM approach.* In Visual Data Mining, pages 281–311. Springer, 2008.

[Navarro 13] María-Dolores Navarro, Roberto Lloréns, Enrique Noé, Joan Ferri & Mariano Alcañiz. *Validation of a low-cost virtual reality system for training street-crossing. A comparative study in healthy, neglected and non-neglected stroke individuals.* Neuropsychological Rehabilitation, no. ahead-of-print, pages 1–22, 2013.

[Nishimura 85] Hitoshi Nishimura, Makoto Hirai, Toshiyuki Kawai, Toru Kawata, Isao Shirakawa & Koichi Omura. *Object modeling by distribution function and a method of image generation.* The Transactions of the Institute of Electronics and Communication Engineers of Japan, vol. 68, no. 4, pages 718–725, 1985.

[Oculus VR] Oculus VR. *Oculus Rift.* Website, -. http ://www.oculusvr.com/.

[Ogi 09] Tetsuro Ogi, Yoshisuke Tateyama & So Sato. *Visual data mining in immersive virtual environment based on 4K stereo images.* In Virtual and Mixed Reality, pages 472–481. Springer, 2009.

[Ouramdane 06] Nassima Ouramdane, Samir Otmane, Frédéric Davesne & Malik Mallem. *FOLLOW-ME : a new 3D interaction technique based on virtual guides and granularity of interaction.* In Proceedings

of the 2006 ACM international conference on Virtual reality continuum and its applications, VRCIA'06, 2006.

[Perrenot 12] Cyril Perrenot, Manuela Perez, Nguyen Tran, Jean-Philippe Jehl, Jacques Felblinger, Laurent Bresler & Jacques Hubert. *The virtual reality simulator dV-Trainer® is a valid assessment tool for robotic surgical skills.* Surgical endoscopy, vol. 26, no. 9, pages 2587–2593, 2012.

[Picon 08] Flavien Picon, Mehdi Ammi & Patrick Bourdot. *Case study of haptic methods for selection on cad models.* In IEEE Virtual Reality Conference, VR'08, 2008.

[Pierce 97] Jeffrey S Pierce, Andrew S Forsberg, Matthew J Conway, Seung Hong, Robert C Zeleznik & Mark R Mine. *Image plane interaction techniques in 3D immersive environments.* In Proceedings of the 1997 symposium on Interactive 3D graphics, 1997.

[Pierce 99] Jeffrey S Pierce, Brian C Stearns & Randy Pausch. *Voodoo dolls : seamless interaction at multiple scales in virtual environments.* In Proceedings of the 1999 symposium on Interactive 3D graphics, 1999.

[Poupyrev 96] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst & Tadao Ichikawa. *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, UIST'96, pages 79–80. ACM, 1996.

[Rautaray 12] Siddharth S Rautaray & Anupam Agrawal. *Real time hand gesture recognition system for dynamic applications.* Int J UbiComp, vol. 3, no. 1, pages 21–31, 2012.

[Renard 10] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand & Anatole Lécuyer. *OpenViBE : an open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments.* Presence : teleoperators and virtual environments, vol. 19, no. 1, 2010.

[Ricci 73] A Ricci. *A constructive geometry for computer graphics.* The Computer Journal, vol. 16, no. 2, pages 157–160, 1973.

[Richardson 02] Magnus JE Richardson & Tamar Flash. *Comparing smooth arm movements with the two-thirds power law and the related segmented-control hypothesis.* The Journal of neuroscience, vol. 22, no. 18, 2002.

[Rosa 10] Daniel Alejandro Winkler Rosa & Hubert Hoffmann Nagel. *Selection Techniques for Dense and Occluded Virtual 3D Environments, Supported by Depth Feedback : Double, Bound and*

Depth Bubble Cursors. In XXIX International Conference of the Chilean Computer Science Society (SCCC), 2010, 2010.

[Rothbaum 95] Barbara Olasov Rothbaum, Larry F. Hodges, Rob Kooper, Dan Opdyke, James S. Williford & Max North. *Virtual reality graded exposure in the treatment of acrophobia : A case report.* Behavior Therapy, vol. 26, no. 3, 1995.

[Ruddle 99] Roy A Ruddle, Stephen J Payne & Dylan M Jones. *The effects of maps on navigation and search strategies in very-large-scale virtual environments.* Journal of Experimental Psychology : Applied, vol. 5, no. 1, 1999.

[Ruiz 09] Jaime Ruiz & Edward Lank. *Effects of target size and distance on kinematic endpoint prediction.* Rapport technique, Technical Report CS-2009-25, University of Waterloo, 2009.

[Ruiz 10] Jaime Ruiz & Edward Lank. *Speeding pointing in tiled widgets : understanding the effects of target expansion and misprediction.* In Proceedings of the 15th international conference on Intelligent user interfaces, IUI'10, 2010.

[Rus-Calafell 13] Mar Rus-Calafell, José Gutiérrez-Maldonado, Cristina Botella & Rosa M Baños. *Virtual Reality Exposure and Imaginal Exposure in the Treatment of Fear of Flying A Pilot Study.* Behavior modification, vol. 37, no. 4, pages 568–590, 2013.

[Sahm 05] Cynthia S Sahm, Sarah H Creem-Regehr, William B Thompson & Peter Willemsen. *Throwing versus walking as indicators of distance perception in similar real and virtual environments.* ACM Transactions on Applied Perception (TAP), vol. 2, no. 1, 2005.

[Schkolne 01] Steven Schkolne, Michael Pruett & Peter Schröder. *Surface drawing : creating organic 3D shapes with the hand and tangible tools.* In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 261–268. ACM, 2001.

[Sharma 12] Mitesh Sharma & Alan Horgan. *Comparison of Fresh-Frozen Cadaver and High-Fidelity Virtual Reality Simulator as Methods of Laparoscopic Training.* World journal of surgery, vol. 36, no. 8, pages 1732–1737, 2012.

[Shaw 94] Chris Shaw & Mark Green. *Two-handed polygonal surface design.* In Proceedings of the 7th annual ACM symposium on User interface software and technology, pages 205–212. ACM, 1994.

[Shneiderman 97] B. Shneiderman. *Direct manipulation for comprehensible, predictable and controllable user interfaces.* In Proceedings of the 2nd international conference on Intelligent user interfaces, 1997.

[Simard 09] Jean Simard, Mehdi Ammi, Flavien Picon & Patrick Bourdot. *Potential field approach for haptic selection.* In Proceedings of Graphics Interface 2009, 2009.

[Song 00] Chang Geun Song, No Jun Kwak & Dong Hyun Jeong. *Developing an efficient technique of Selection and Manipulation in Immersive VE.* In Proceedings of the ACM symposium on Virtual reality software and technology, VRST'00, 2000.

[Sony] Sony. *HMZ-T3W Personal 3D Viewer.* Website, -. www.sony.co.uk/product/personal-3d-viewer/hmz-t3w.

[Stanney 95] Kay Stanney. *Realizing the full potential of virtual reality : human factors issues that could stand in the way.* In Proceedings of Virtual Reality Annual International Symposium, pages 28–34. IEEE, 1995.

[Steed 04] Anthony Steed & Chris Parker. *3d selection strategies for head tracked and non-head tracked operation of spatially immersive displays.* In 8th International Immersive Projection Technology Workshop, 2004.

[Steinicke 08] Frank Steinicke, Klaus Hinrichs, Johannes Schöning & Antonio Krüger. *Multi-touching 3D data : Towards direct interaction in stereoscopic display environments coupled with mobile devices.* In Advanced Visual Interfaces Workshop on Designing Multi-Touch Interaction Techniques for Coupled Public and Private Displays, AVI'08, pages 46–49. Citeseer, 2008.

[Stoakley 95] Richard Stoakley, Matthew J Conway & Randy Pausch. *Virtual reality on a WIM : interactive worlds in miniature.* In Proceedings of the SIGCHI conference on Human factors in computing systems, 1995.

[Teather 09] Robert J Teather, Andriy Pavlovych, Wolfgang Stuerzlinger & I Scott MacKenzie. *Effects of tracking technology, latency, and spatial jitter on object movement.* In 2009 IEEE Symposium on 3D User Interfaces, 3DUI'2009, 2009.

[Teather 11] Robert J Teather & Wolfgang Stuerzlinger. *Pointing at 3D targets in a stereo head-tracked virtual environment.* In 2011 IEEE Symposium on 3D User Interfaces, 3DUI'2011, 2011.

[Tonnis 05] Marcus Tonnis, Christian Sandor, Christian Lange & Heiner Bubb. *Experimental evaluation of an augmented reality visualization for directing a car driver's attention.* In Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR'05, 2005.

[Tonnis 06] Marcus Tonnis & Gudrun Klinker. *Effective control of a car driver's attention for visual and acoustic guidance towards the*

direction of imminent dangers. In Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR'06, 2006.

[Torres-Solis 09] Jorge Torres-Solis. *Mediated reality and location awareness to facilitate topographical orientation.* PhD thesis, University of Toronto, 2009.

[Trueba Hornero 10] Ramón Trueba Hornero, Carlos Antonio Andújar Gran, Fernando Argelaguet Sanz*et al. World-in-miniature interaction for complex virtual environments.* International Journal of Creative Interfaces and Computer Graphics, vol. 1, no. 2, 2010.

[Tsingos 95] Nicolas Tsingos, Eric Bittar & Marie-Paule Gascuel. *Implicit surfaces for semi-automatic medical organ reconstruction.* Computer Graphics International'95, pages 3–15, 1995.

[Ullah 08] Sehat Ullah, Nassima Ouramdane, Samir Otmane, Paul Richard, Frédéric Davesne & Malik Mallem. *Augmenting 3d interactions with haptic guide in a large scale virtual environment.* In Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry, VRCAI'08, 2008.

[Usoh 99] Martin Usoh, Kevin Arthur, Mary C Whitton, Rui Bastos, Anthony Steed, Mel Slater & Frederick P Brooks Jr. *Walking> walking-in-place> flying, in virtual environments.* In Siggraph, volume 99, pages 359–364, 1999.

[Vanacken 07] Lode Vanacken, Tovi Grossman & Karin Coninx. *Exploring the effects of environment density and target visibility on object selection in 3D virtual environments.* In IEEE Symposium on 3D User Interfaces, 2007, 3DUI'07, 2007.

[Vanacken 09] Lode Vanacken, Tovi Grossman & Karin Coninx. *Multimodal selection techniques for dense and occluded 3D virtual environments.* International Journal of Human-Computer Studies, vol. 67, no. 3, pages 237–255, 2009.

[Våpenstad 13] Cecilie Våpenstad, Erlend Fagertun Hofstad, Thomas Langø, Ronald Mårvik & Magdalena Karolina Chmarra. *Perceiving haptic feedback in virtual reality simulators.* Surgical endoscopy, pages 1–7, 2013.

[Veit 11] Manuel Veit, Antonio Capobianco & Dominique Bechmann. *An experimental analysis of the impact of touch screen interaction techniques for 3-d positioning tasks.* In Virtual Reality Conference, VR'11, pages 75–82. IEEE, 2011.

[Veit 12] Manuel Veit, Antonio Capobianco & Dominique Bechmann. *CrOS : a touch screen interaction technique for cursor manipu-*

lation on 2-manifolds. In Proceedings of the 18th ACM symposium on Virtual reality software and technology, pages 97–100. ACM, 2012.

[Virtuix 13] Virtuix. *Virtuix Omni^{TM}.* Website, 2013. http ://www.virtuix.com/.

[Walker 93] Neff Walker, David E Meyer & JOhn B Smelcer. *Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction.* Human Factors : The Journal of the Human Factors and Ergonomics Society, vol. 35, no. 3, pages 431–458, 1993.

[Wang 04] W. Wang. *Human Navigation Performance Using 6 Degree of Freedom Dynamic Viewpoint Tethering in Virtual Environments.* PhD thesis, University of Toronto, Mechanical and Industrial Engineering, 2004.

[Wang 12] Jia Wang & Rob Lindeman. *Leaning-based travel interfaces revisited : frontal versus sidewise stances for flying in 3D virtual spaces.* In Proceedings of the 18th ACM symposium on Virtual reality software and technology, VRST'12, pages 121–128. ACM, 2012.

[Ware 94] Colin Ware & Ravin Balakrishnan. *Reaching for objects in VR displays : lag and frame rate.* ACM Transactions on Computer-Human Interaction (TOCHI), vol. 1, no. 4, 1994.

[Wessberg 00] Johan Wessberg, Christopher R Stambaugh, Jerald D Kralik, Pamela D Beck, Mark Laubach, John K Chapin, Jung Kim, S James Biggs, Mandayam A Srinivasan & Miguel AL Nicolelis. *Real-time prediction of hand trajectory by ensembles of cortical neurons in primates.* Nature, vol. 408, no. 6810, 2000.

[Willemsen 08] Peter Willemsen, Amy A Gooch, William B Thompson & Sarah H Creem-Regehr. *Effects of stereo viewing conditions on distance perception in virtual environments.* Presence : Teleoperators and Virtual Environments, vol. 17, no. 1, 2008.

[Wingrave 05] C Wingrave & D Bowman. *Baseline factors for raycasting selection.* In Proceedings of HCI International, 2005.

[Woodworth 99] Robert Sessions Woodworth. *Accuracy of voluntary movement.* The Psychological Review : Monograph Supplements, vol. 3, no. 3, 1899.

[Yamada 02] Toshio Yamada, Daisuke Tsubouchi, Tetsuro Ogi & Michitaka Hirose. *Desk-sized immersive workplace using force feedback grid interface.* In Proceedings IEEE Virtual Reality 2002, VR'02, 2002.

[Zeleznik 96] Robert C Zeleznik, Kenneth P Herndon & John F Hughes. *SKETCH : An Interactive for Sketching 3D scenes.* In Proceedings of ACM SIGGRAPH, volume 96, 1996.

[Zellweger 03] Polle T Zellweger, Jock D Mackinlay, Lance Good, Mark Stefik & Patrick Baudisch. *City lights : contextual views in minimal space.* In Proceedings ACM Human factors in computing systems, CHI'03, 2003.

[Zhai 94] Shumin Zhai, William Buxton & Paul Milgram. *The "Silk Cursor" : investigating transparency for 3D target acquisition.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 459–464. ACM, 1994.

[Zhai 97] Shumin Zhai & John W Senders. *Investigating coordination in multidegree of freedom control I : time-on-target analysis of 6 DOF tracking.* In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pages 1249–1253. SAGE Publications, 1997.

Bibliographie personnelle

[Wonner 10a] Jonathan Wonner. Aide haptique à la sélection de cibles dans un nuage de points en réalité virtuelle. Master's thesis, Université de Rennes 1, 2010.

[Wonner 10b] Jonathan Wonner, Jérôme Grosjean, Antonio Capobianco & Dominique Bechmann. *Aide haptique prédictive à la sélection de cibles en trois dimensions.* In Cinquièmes Journées de l'Association Française de la Réalité Virtuelle AFRV, 2010.

[Wonner 11a] Jonathan Wonner. *Aides à la sélection de cibles en réalité virtuelle.* Poster, 2011. Journée Poster de l'École Doctorale MSII.

[Wonner 11b] Jonathan Wonner. *STARFISH : a new assistance to select targets in virtual reality*, 2011. Workshop franco-japonais Next Generation of Virtual Reality, Todai Forum.

[Wonner 11c] Jonathan Wonner, Jérôme Grosjean, Antonio Capobianco & Dominique Bechmann. *SPEED : Prédiction de cibles.* In 23rd French Speaking Conference on Human-Computer Interaction, page 19. ACM, 2011.

[Wonner 12a] Jonathan Wonner. *Starfish : a selection technique for dense virtual environment*, 2012. CfP 3rd Sino-French Symposium on Computer Graphics and Virtual Reality.

[Wonner 12b] Jonathan Wonner, Jérôme Grosjean, Antonio Capobianco & Dominique Bechmann. *Starfish : a selection technique for dense virtual environments.* In Proceedings of the 18th ACM symposium on Virtual reality software and technology, pages 101–104. ACM, 2012.

[Wonner 13] Jonathan Wonner, Jérôme Grosjean, Antonio Capobianco & Dominique Bechmann. *Bubble bee, an alternative to arrow for pointing out directions.* In Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology, pages 97–100. ACM, 2013.

Chapitre A

Carré latin

Lors d'un protocole expérimental, nous souhaitons éliminer, ou au moins réduire, les différents effets d'apprentissage entre les conditions. Par exemple, dans notre expérience d'évaluation de Starfish du chapitre 6, si un utilisateur manipule d'abord Depth Ray puis Starfish, il peut ne pas obtenir les mêmes résultats que s'il manipule d'abord Starfish puis Depth Ray. Plusieurs facteurs sont en cause, comme la fatigue physique liée à la durée de l'expérience, ou au fait que la compréhension de la scène pour l'utilisateur s'améliore au cours de la manipulation. Si tous les utilisateurs effectuent toutes les tâches demandées dans le même ordre, les résultats obtenus sont influencés par cet ordre. Afin de réduire cet effet, nous permutons les conditions présentées entre les participants. Chaque permutation, correspondant donc à un ordre pour les conditions, devrait être affectée à un même nombre de participants, pour ne pas « avantager » l'une d'entre elles. Il est toutefois rare que toutes les permutations soient considérées. Dans l'évaluation de Starfish, nous avons deux paramètres, la technique et la densité de la scène, pouvant prendre chacun trois valeurs. Nous avons donc au total neuf (3×3) conditions possibles, pour un total de 362.880 permutations (9!). Ayant des difficultés pour réunir autant de participants, nous décidons de changer nos critères de la manière suivante. Un participant donné effectue toutes les tâches pour une technique donnée, puis toutes les tâches pour une autre technique et enfin toutes les tâches pour la dernière technique. Cela réduit le nombre de permutations à 1296. En effet, nous avons 9 choix possibles pour la première condition qui fixe donc la première technique utilisée, puis 2 choix possibles pour la deuxième condition, puis 1 choix possible pour la troisième condition. Il reste 6 choix possibles pour la quatrième condition, qui fixe la deuxième technique utilisée, et ainsi de suite. Ce nombre étant encore trop élevée, nous décidons de mélanger aléatoirement, pour une technique donnée, l'ordre des trois densités présentées. Cette décision nous laisse 6 permutations possibles. Pour des raisons de simplicité, nous choisissons de ne garder que trois permutations circulaires. Ce type de raisonnement est simplifié par l'utilisation d'un carré latin.

Un carré latin d'ordre n, pour $n \in \mathbb{N}$, est une matrice carrée de taille $n \times n$ contenant n éléments distincts, généralement les entiers compris entre 0 et $n-1$, de telle sorte que chaque ligne et chaque colonne ne contiennent qu'un seul exemplaire de chacun de

ces n éléments. Une grille de Sudoku est par exemple un cas particulier de carré latin d'ordre 9. Dans notre contexte, chaque élément de la grille est une condition (comme une technique de sélection), et chaque colonne une permutation de ces conditions. En reprenant les notations du chapitre 6, nous avons le carré latin d'ordre 3 suivant :

C_0	C_1	C_2
DR	SFU	SFA
SFU	SFA	DR
SFA	DR	SFU

Au premier participant U_0 de notre expérience est attribuée la permutation représentée par la colonne C_0. Plus généralement, au participant U_k est attribuée la permutation de la colonne $C_{k \mod 3}$.

Chapitre B

Méthode des moindres carrés

Dans le chapitre 7, nous souhaitons modéliser un ensemble de données expérimentales par une courbe quadratique. La méthode des moindres carrés est bien adaptée à cette tâche [Legendre 05]. En effet, cette méthode permet d'obtenir la fonction décrivant le « mieux » des données, parmi une famille de fonctions. Nous l'utilisons dans le cas où cette famille de fonction est l'ensemble des polynômes de degré 2.

Soit N le nombre de données expérimentales disponibles. Notons (x_i, y_i) la donnée i, pour $i \in [\![1; N]\!]$. Pour tout triplet $(a, b, c) \in \mathbb{R}^3$, posons la fonction $f_{a,b,c} : \mathbb{R} \longrightarrow \mathbb{R}$, définie par :

$$\forall x \in \mathbb{R}, f_{a,b,c}(x) = ax^2 + bx + c$$

La dérivée de $f_{a,b,c}$ s'exprime donc :

$$\forall x \in \mathbb{R}, f'_{a,b,c}(x) = 2ax + b$$

Nous cherchons à minimiser l'erreur quadratique entre les données expérimentales et la fonction $f_{a,b,c}$. Cette erreur est définie par la somme suivante :

$$S(a, b, c) = \sum_{i=1}^{N} \left(y_i - f_{a,b,c}(x_i)\right)^2$$

Minimiser cette valeur revient à chercher les valeurs de a, b et c telles que les dérivées partielles de S s'annulent :

$$\frac{\partial S}{\partial a}(a, b, c) = \frac{\partial S}{\partial b}(a, b, c) = \frac{\partial S}{\partial c}(a, b, c) = 0$$

Pour notre algorithme SPEED, nous simplifions ce problème par deux contraintes. Nous souhaitons en effet que la courbe quadratique passe par le pic de vitesse, et que ce pic représente l'extremum de la courbe. Nous considérons toutes les données enregistrées depuis le pic de vitesse. Notons (x_1, y_1) la donnée correspondant à ce pic (la première de notre ensemble de données). Nos contraintes s'expriment de la manière suivante :

$$f_{a,b,c}(x_1) = ax_1^2 + bx_1 + c = y_1$$
$$f'_{a,b,c}(x_1) = 2ax_1 + b = 0$$

De ces équations, nous obtenons facilement :

$$b = -2ax_1$$
$$c = y_1 + ax_1^2$$

Notre fonction $f_{a,b,c}$ peut donc se réécrire à l'aide du seul paramètre a :

$$\begin{aligned} f_a(x) &= ax^2 - 2ax_1x + y_1 + ax_1^2 \\ &= a(x - x_1)^2 + y_1 \end{aligned}$$

De la même manière, nous pouvons réécrire l'erreur quadratique comme une fonction de a :

$$S(a) = \sum_{i=1}^{N} \Big(a(x_1 - x_i)^2 + y_1 - y_i \Big)^2$$

Une simple dérivation par le paramètre a donne :

$$S'(a) = \sum_{i=1}^{N} 2(x_1 - x_i)^2 \Big(a(x_1 - x_i)^2 + y_1 - y_i \Big)$$

Nous souhaitons trouver la valeur de a pour laquelle $S'(a) = 0$, soit :

$$\sum_{i=1}^{N} 2(x_1 - x_i)^2 \Big(a(x_1 - x_i)^2 + y_1 - y_i \Big) = 0.$$

En séparant les membres contenant le terme a, nous arrivons à :

$$a \sum_{i=1}^{N} (x_1 - x_i)^4 = - \sum_{i=1}^{N} (x_1 - x_i)^2 (y_1 - y_i)$$

Finalement, nous obtenons les paramètres de notre courbe quadratique :

$$a = - \frac{\sum_{i=1}^{N} (x_1 - x_i)^2 (y_1 - y_i)}{\sum_{i=1}^{N} (x_1 - x_i)^4}$$
$$b = -2ax_1$$
$$c = y_1 + ax_1^2$$