Sur le contexte spatial en gestion des fenêtres et interaction homme-machine.

Guillaume Faure

UNIVERSITÉ PARIS SUD XI

ÉCOLE DOCTORALE D'INFORMATIQUE
DE PARIS SUD

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université Paris Sud XI
Mention : INFORMATIQUE

Présentée et soutenue par

GUILLAUME FAURE

SUR LE CONTEXTE SPATIAL EN GESTION DES FENÊTRES ET INTERACTION HOMME-MACHINE

soutenue le 15 Décembre 2011

Encadrant de thèse: Olivier Chapuis

Directeur de thèse: Michel Beaudouin-Lafon

Jury :

Emmanuel Dubois	Rapporteur
Patrick Girard	Rapporteur
Géry Casiez	Examinateur
Chantal Reynaud	Examinateur
Olivier Chapuis	Encadrant
Michel Beaudouin-Lafon	Directeur de thèse

Remerciements

Je tiens tout d'abord à remercier mes encadrants, Michel Beaudouin-Lafon et Olivier Chapuis pour leur soutient, leurs remarques toujours constructives et généralement pour avoir su me guider pendant les quatre dernières années. Ce sont tous deux des chercheurs de premier ordre, passionnés par ce qu'ils font. Par leurs conseils et leurs encouragements, ils ont su m'épauler dans les moments difficiles que tous étudiant rencontre lors de son doctorat.

Je remercie bien évidemment tous les membres de mon jury qui ont accepté de lire mes travaux et de s'être déplacés parfois de loin pour assister à ma soutenance. Merci en particulier à mes rapporteurs Émmanuel Dubois et Patrick Girad pour toutes leurs remarques justifiées.

Je voudrais également remercier tous les membres de l'équipe insitu et tout particulièrement Stéphane Huot et Caroline Appert qui ont su également m'aider dans les moments difficiles et proposer leurs points de vue éclairant tout au long de ces années d'étude. Bien sur je remercie chaleureusement mes co-étudiants, les différents ingénieurs de recherche et toutes les personnes qui ont su me supporter, certaines pendant longtemps et ont toujours entretenues une ambiance de travail studieuse, mais détendu, avec beaucoup de rires et de grands moments. Merci aussi à Wendy Mackay pour son point de vue, toujours intéressant et son implication constante dans la vie de l'équipe insitu.

Je voudrais également exprimer mon immense gratiture envers toute ma famille qui a toujours été derrière moi pour m'encourager ou me pousser toujours avant. Merci également à mes amis Marie-Émilie, Frédéric, Geoffroy, Rémy, Xavier, Katelle, Thomas, Alexandre et bien d'autres pour leur participation et leur soutient.

Enfin, merci tout spécialement à Bo Kyung.

Table des matières

Résumé **xiii**

1 Introduction & Motivations **1**
1.1 La Notion de Fenêtre . . . 2
1.2 Gestion de fenêtres . . . 5
1.2.1 Allocation de l'Espace d'Affichage . . . 5
Alternatives au Chevauchement de Fenêtres . . . 5
La Tâche au Cœur de la Gestion de Fenêtres . . . 7
Contourner le Chevauchement . . . 9
Autres Solutions . . . 10
1.2.2 Sélection et Changement de Fenêtre . . . 12
1.3 Spatialité, Stabilité et Localité d'Interaction . . . 14
1.4 Le problème . . . 15

2 Power Tools **19**
2.1 Introduction . . . 19
2.2 Interaction avec des Historiques de Sélection, de Déplacement et de Copie . . . 21
2.3 Comparaison de TimeShit à d'autres mécanisme d'historiques . . . 25
2.4 DeskPop . . . 31
2.5 StackLeafing . . . 33
2.6 Couches de Fenêtres . . . 38
2.7 Implémentation . . . 39
2.8 Conclusion . . . 40

3 Pointage sur Cibles Animées ou Surgissantes **41**
3.1 Introduction . . . 41
3.2 Motivations dans le cadre de la thèse . . . 45
3.3 Expérience . . . 47
3.3.1 Matériel . . . 47
3.3.2 Participants . . . 47
3.3.3 Tâche et procédure . . . 47
3.3.4 Protocole . . . 49
3.4 Résultats . . . 50
3.5 Analyse Cinématique . . . 54
3.6 Étendre la loi de Fitts . . . 56
3.7 Conclusion . . . 59

4 Perception de la Profondeur **61**
4.1 Introduction . . . 61
4.2 La tâche . . . 63
4.3 Sujets et Matériel Utilisé . . . 64
4.4 Protocole Expérimental . . . 64
4.5 Prédictions . . . 65
4.6 Résultats . . . 65
4.7 Conclusion et Travaux Futurs . . . 69

5 Interaction Rythmique **71**
5.1 Introduction . . . 71
5.2 Motivations . . . 72
5.2.1 Avantages du Rythme comme Méthode d'Entrée . . . 73
5.2.2 Utilisation des Motifs Rythmiques . . . 73
5.3 État de l'art . . . 75
5.4 Des Motifs Rythmiques pour l'Interaction . . . 76
5.5 Expérience 1 : Reproduction de Motifs Rythmiques . . . 79
5.5.1 Le Reconaisseur . . . 79
5.5.2 Matériel et Participants . . . 80
5.5.3 Stimulus . . . 80
5.5.4 Feedbacks Utilisateur . . . 81
5.5.5 Vocabulaire . . . 82
5.5.6 La Tâche . . . 83
5.5.7 Procédure et Schéma Expérimental . . . 83
5.5.8 Résultats Quantitatifs . . . 84
5.5.9 Résultats Qualitatifs . . . 87
5.6 Outil d'Analyse : AnEwe . . . 87
5.7 Un Classifieur de Motifs . . . 91
5.8 Expérience 2 : Mémorisation des motifs rythmiques . . . 94
5.8.1 Variables . . . 94
5.8.2 La Tâche . . . 94
5.8.3 Apparatus & Participants . . . 97
5.8.4 Procédure et schéma expérimental . . . 97
5.8.5 Résultats Quantitatifs . . . 98
5.8.6 Résultats Qualitatifs . . . 100
5.9 Conclusion . . . 101

6 Conclusion et Perspectives **103**
6.1 Stabilité Spatiale . . . 103
6.1.1 Le problème . . . 103
6.1.2 Les Contributions . . . 104
6.1.3 Les Perspectives . . . 106
6.2 Localité de l'Interaction . . . 107
6.2.1 Le Problème . . . 107

6.2.2 Les Contributions 108
6.2.3 Les Perspectives 109

Bibliographie **111**

Table des figures

1 Plan de la thèse. xiv

Introduction & Motivations **1**
1.1 Le Xerox Star 3
1.2 Apple Lisa et Mac OS. 4
1.3 Elastic Windows 6
1.4 La Task Gallery [98] 8
1.5 QT Inner windows 10
1.6 Panorama dans Firefox 11
1.7 Piles, pliages et enroulement 13
1.8 Exposé d'Apple 14
1.9 Taskposé 15

Power Tools **19**
2.1 TimeShift 23
2.2 Interactions temporisées 24
2.3 Microsoft Office Clipboard 28
2.4 Klipper 29
2.5 La technique d'interaction DeskPop 32
2.6 Gestes de Fold n' Drop 35
2.7 Exemple d'utilisation de *StackLeafing* 36
2.8 Trailing Widget 37
2.9 L'algorithme de création des couches de fenêtres. 38

Pointage sur Cibles Animées ou Surgissantes **41**
3.1 Boîte de dialogue du navigateur internet Firefox 43
3.2 Déroulement d'une épreuve de l'expérience de pointage 48
3.3 Influence de la distance et de la largeur de la cible sur le temps de pointage 52
3.4 Influence du délai d'animation/d'apparition de la cible sur le temps de pointage 52
3.5 Influence du délai d'animation/d'apparition de la cible sur le temps de premier mouvement et le nombre de sous-mouvements 54
3.6 Vitesse moyenne du mouvment de pointage en fonction de la distance pour une distance de 768 55
3.7 Regressions linéaires des données de l'expérience de pointage : la loi de Fitts et le modèle modifié 58

Perception de la Profondeur **61**
4.1 Les trois indices de profondeur testés 63
4.2 Influence de l'indice visuel de profondeur sur le taux d'erreur . . . 67
4.3 Influence de l'indice visuel de profondeur sur le temps de réponse 67
4.4 Influence du nombre de fenêtres et de couches sur le taux d'erreur 68
4.5 Influence de l'interaction entre le l'indice visuel de profondeur et le nombre de couches de fenêtres sur le taux d'erreur 69

Interaction Rythmique **71**
5.1 Les seize motifs rythmiques à trois pulsations possibles 78
5.2 Différence entre liste d'événement et structure rythmique 80
5.3 Stimulus de l'expérience de reproduction des motifs rythmiques . 81
5.4 Feedback visuel lors de la reproduction d'un motif par l'utilisateur pour l'expérience de reproduction de motifs rythmiques. 81
5.5 Le vocabulaire utilisé lors de l'expérience de reproduction des motifs rythmiques. 82
5.6 Exemples d'erreurs de reproduction de motifs rythmiques 84
5.7 Influence du feedback utilisateur sur le taux de reconnaissance (reconnaisseur stricte) 85
5.8 Influence du nombre d'événements et de la longueur des motifs sur le taux de reconnaissance (reconnaisseur stricte) 85
5.9 Influence du nombre d'inversion sur le taux de reconnaissance (reconnaisseur stricte) 86
5.10 AnEwe : un outil d'aide à la conception de reconnaisseur 88
5.11 Outil de filtre dans AnEwe 90
5.12 Exploration des résultats à l'aide d'AnEwe 90
5.13 Distributions agrégées générées par AnEwe 90
5.14 Algorithme du classifieur de motifs. 92
5.15 Influence du feedback utilisateur sur taux de reconnaissance (classifieur) 92
5.16 Influence du nombre de tapes et de la longueur des motifs sur le taux de reconnaissance (classifieur) 93
5.17 Commandes de l'expérience de mémorisation des motifs rythmiques 95
5.18 Stimulus de la phase d'apprentissage de l'expérience de mémorisation 96
5.19 Écrans de confirmation dans l'expérience sur la mémorisation des raccourcis rythmiques 96
5.20 Plan expérimental pour l'étude de la mémorisation des raccourcis rythmiques 98
5.21 Taux de mémorisation des raccourcis rythmiques 98
5.22 Influence de la sous-session sur le taux d'utilisation de l'aide lors de l'expérience de mémorisation 99
5.23 Taux d'utilisation des raccourcis rythmiques lors de la phase libre 100

Liste des tableaux

Introduction & Motivations **1**

Power Tools **19**
2.1 Valeur utilisées lors des analyses KLM 26
2.2 Analyse KLM de Microsoft Office Clipboard 27
2.3 Analyse KLM de Klipper (clavier) 29
2.4 Analyse KML de Klipper (souris) 30
2.5 Analyse KLM de TimeShift 30

Pointage sur Cibles Animées ou Surgissantes **41**
3.1 Vue d'ensemble des résultat de l'expérience de pointage 51
3.2 Paramètres de la loi de Fitts 57

Perception de la Profondeur **61**
4.1 Analyse statistique du temps de réponse 66
4.2 Analyse statistiques du taux d'erreurs 66

Interaction Rythmique **71**

Conclusion et Perspectives **103**

Résumé

Lors de l'usage d'un système informatique, l'utilisateur dispose, en plus des connaissances requises pour mener à bien sa tâche principale, d'un ensemble d'informations n'ayant pas trait à cette dernière. Ces informations n'en sont pas moins importantes, puisqu'elles participent à la réalisation de la tâche en permettant une manipulation plus aisée des outils nécessaires à sa réalisation. Il est communément admis qu'une interaction doit être conçue "en contexte", c'est-à-dire quelle doit prendre en compte cet ensemble, mal définit, d'informations que nous appelons pour plus de facilité le *contexte*. Le but de mes travaux a été d'étudier et de comprendre la partie spatiale de ce contexte afin de l'intégrer lors de la conception de nouvelles interactions.

Le support choisi initialement dans cette thèse pour l'étude du *Contexte Spatial* est la navigation dans l'interface graphique utilisateur d'un système informatique. À l'heure actuelle, la majorité des interfaces utilisateur dérivent encore de la métaphore du bureau virtuel. Cette famille d'interfaces est apparue dès le début des années 80 avec le Xerox Alto puis le Xerox Star. Son adoption par l'informatique grand public, notamment avec Apple (Lisa, Macintosh) puis par Microsoft avec son système d'exploitation Windows (3.11) a fini d'en faire un standard de fait. Ce type d'environnement adopte le paradigme dit Windows, Icons, Menus and Pointing (WIMP). Les fenêtres sont des zones graphiques que l'on peut redimensionner et pouvant se chevaucher. Ces fenêtres offrent un espace d'affichage séparé et indépendant pour chacune des applications en cours d'exécutions ou des documents ouverts. La fenêtre est l'objet d'interaction de base, allant de l'application complète à une simple boîte de dialogue. La sélection d'une fenêtre par l'utilisateur lui permet également d'indiquer au système avec quelle application il veut désormais interagir. Les fenêtres permettent donc l'exécution concurrente de plusieurs programmes en multiplexant les entrées et les sorties d'un ordinateur. Ce que l'on appelle navigation dans l'interface est ici l'ensemble des actions que doit effectuer l'utilisateur afin d'accéder à différents éléments de l'interface et à leurs fonctionnalités. Un exemple classique de navigation est celui du changement de fenêtre qui a été largement étudié dans la communauté IHM.

Toutefois, ce type d'interface commence à montrer ses limites face au nombre toujours croissant de contenus à gérer. Lors de l'introduction de ces environnements de bureaux virtuels, la puissance des machines et les possibilités réduites offertes par les logiciels existants limitaient le nombre de fenêtres affichées simultanément. Cependant, la puissance de calcul disponible a augmenté, les usages de l'outil informatique se sont diversifiés et l'espace d'affichage disponible s'est accru et complexifié (multiple surfaces d'affichage). Les utilisateurs sont de ce fait plus à même de mener plusieurs tâches de front sur un même système et à conserver

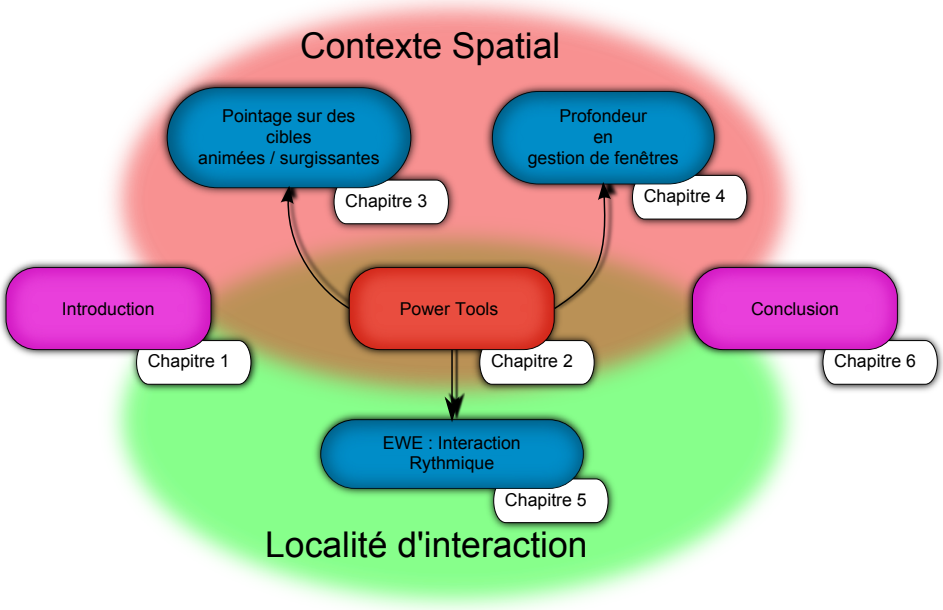


FIGURE 1 – *Plan de la thèse.*

plus de fenêtres affichées simultanément. Ces évolutions ont entrainé la modification progressive des habitudes de gestion de fenêtres. Par exemple, de moins en moins d'utilisateurs dimensionnent systématiquement leurs fenêtres de façon à couvrir entièrement l'espace disponible [60] (maximiser). Enfin, les applications elles-mêmes se sont enrichies introduisant de nouveaux moyens de gérer de multiples documents à l'intérieur d'une même fenêtre (onglets, vues séparées, espaces de docking...) [1]. Le navigateur web en est un exemple flagrant avec l'introduction des onglets puis plus récemment de techniques pour gérer ces onglets comme Panorama de Mozilla. L'ensemble de ces facteurs fait de la navigation dans l'interface utilisateur une tâche complexe et coûteuse en temps.

Motivé par le but de faciliter cette navigation, nous avons imaginé trois techniques d'interaction regroupées sous l'appellation *Power Tools* et constituant le chapitre 2 de ce document (voir Figure 1) :

Desk Pop : Une interaction permettant d'interagir avec le contenu du bureau (typiquement des icônes) tout en conservant les fenêtres visibles à leur emplacement d'origine. Contrairement aux interactions classiques (*Show Desktop*, *Exposé*), qui modifient la disposition des fenêtres à l'écran afin de donner accès au bureau, cette technique se contente de placer le bureau au dessus des fenêtres puis d'appliquer un effet visuel permettant de voir les fenêtres à travers le fond du bureau.

1. voir par exemple http ://dubroy.com/blog/2009/01/29/my-talk-at-mozilla/.

Stack Leafing : Une interaction permettant de parcourir les fenêtres (et le bureau) lors d'une opération de glisser-déposer afin de découvrir la zone de dépôt. L'originalité de cette technique réside dans le mode de parcours des fenêtres qui ne se fait plus individuellement, mais par groupe de fenêtres appelées "couche". Ces couches sont constitués de fenêtres ne se recouvrant pas, permettant ainsi d'en visualiser la totalité.

Time Shift : Une interaction pour accéder à l'historique de toutes les opérations de copie ou de déplacement (texte, fichier...) de l'utilisateur.

Les deux premières interactions ont été conçues pour faciliter la navigation entre les fenêtres (et le bureau) lors d'une opération de glisser-déposer : une opération réputée difficile du fait du pseudo-mode introduit par la pression continue du bouton de la souris, empêchant d'utiliser les interactions usuelles. Tout au long de la conception de ces techniques d'interaction, un grand soin a été pris afin de conserver la disposition des objets à l'écran. La conception et l'implémentation de ces nouvelles techniques d'interaction a été riche en questions et en enseignements :

– Qu'est-ce exactement que le contexte spatial dans le cadre des gestionnaires de fenêtres permettant le recouvrement de fenêtres ?
– La modification de ce contexte spatial lors d'une interaction affecte-t-elle les utilisateurs ?

Afin de répondre à ces questions, nous avons mené deux études plus théoriques, appuyées par des expériences utilisateurs contrôlées en laboratoire. Ces études, bien que directement inspirées des travaux rassemblés dans la première partie de cette thèse (*Power Tools* chapitre 2), et ayant pour but premier de valider et de mieux comprendre les choix effectués ont été conçues dans une optique plus large. Les tâches utilisées lors de ces expériences utilisateurs ont été conçues pour être à la fois simples, afin de limiter le nombre de facteurs, et abstraites, par souci de généralisation.

La première étude (formant le troisième chapitre de ce document, voir Figure 1) s'est intéressée à la validation du design de l'interaction *Desk Pop*. Le but de cette interaction est de conserver le contexte spatial de l'utilisateur lors de l'interaction avec un bureau recouvert par des fenêtres. L'expérience modélisait l'interaction, à l'aide des deux techniques standard ainsi qu'à l'aide de notre technique *Desk Pop*, d'un utilisateur exécutant une opération de glisser-déposer d'une icône originellement recouverte se trouvant sur le bureau vers une des fenêtres ouvertes. Cependant, en abstrayant autant que faire se peut la tâche, cette expérience a été réalisée de façon à ce que ses résultats soient généralisables à toute tâche de pointage sur une cible animée ou surgissante lors du mouvement de pointage. Conceptuellement, cette expérience étudie l'impact du changement du contexte spatial lors d'un pointage. Dans une certaine mesure, elle recherche également l'existence

d'un contexte spatial "caché" que les utilisateurs conserveraient en mémoire même lorsque les éléments visuels ont disparu.

La seconde étude (formant le quatrième chapitre de ce document, voir Figure 1) portait sur la technique *Stack Leafing*. Cette technique propose de ne plus explorer les fenêtres individuellement, mais par "couches". Nous avons donc voulu matérialiser ces couches de fenêtres en utilisant des aides visuelles (luminosité, flou, ombre) servant à indiquer la profondeur à laquelle se trouvent ces couches. Nous avons conduit une expérience pour évaluer la pertinence de ces différents "indices visuels de profondeur". La profondeur a toujours été une dimension assez difficile à comprendre et à conceptualiser en gestion de fenêtres. On parle souvent de scène en 2 dimensions et demi.

Les Power Tools n'ont pas introduit que des questions d'ordre théoriques sur le contexte spatial. La seconde partie de cette thèse se penche sur problème de l'appel de commandes (ou l'activation d'interactions). Une question est apparue lors de la conception des Power Tools : pourquoi les utilisateurs n'utilisent pas les systèmes d'historique déjà existants pour le copier-coller ? Une cause possible est la complexité requise afin d'appeler ces historiques, ces interactions forçant les utilisateurs à détourner leur attention loin de leur point d'intérêt ou d'attention original, par exemple lors de l'utilisation d'un menu. Power Tools intègre donc des interactions au système en augmentant, à l'aide de la dimension temporelle, les interactions classiques déclenchant les opérations augmentées par nos historiques. Cette dimension temporelle est apparue comme prometteuse dans le cadre de la conception d'interfaces conservant la localité d'interaction. La dernière partie de cette thèse a donc étudié, en collaboration avec Emilien Ghomi, l'utilisation de rythmes comme méthode d'entrée.

Cette thèse part d'une idée simple : conserver la disposition des objets à l'écran afin de limiter l'effort mental requis par l'utilisateur pour se repérer dans l'interface. Suivant cette idée directrice, nous avons conçu une collection d'interactions permettant la navigation lors d'un glisser-déposer et d'accéder à des historiques d'interaction ce qui nous a permis de construire des intuitions sur les propriétés importantes de telles interactions. Par la suite, il était alors nécessaire de valider ces intuitions par des expériences utilisateur. Nous avons choisi de réaliser des expériences en laboratoire sur des abstractions de ces techniques afin de fournir des conclusions plus générales. En conséquence, nos observations permettent de capturer des phénomènes intervenant dans d'autres situations que la gestion des fenêtres. Et ainsi, en contribuant à une meilleure compréhension des phénomènes, nous pensons que ses résultats peuvent rationaliser ou inspirer la conception de nouvelles techniques d'interaction.

Les travaux effectués au cours de cette thèse ont fait l'objet de quatre publications : deux articles dans des conférences internationales, ACM CHI [37] et INTERACT [35], un article dans la conférence nationale IHM [36], et une présenta-

tion au consortium doctoral de cette même conférence IHM [34]. La dernière partie de cette thèse sur l'interaction rythmique vient d'être acceptée à CHI 2012 [43].

CHAPITRE 1

Introduction & Motivations

L'utilisation des premiers ordinateurs était réservée à des personnes fortement spécialisées. L'entrée d'informations dans le système était une gageure et toute action sur le système pouvait être considérée comme un acte de programmation. Dans le meilleur des cas, l'utilisateur crée un programme qu'il encode sur un support (des cartes perforées par exemple) qu'il donne ensuite à un technicien chargé de le faire exécuter par la machine. L'utilisateur récupère, le plus souvent le lendemain, les résultats de l'exécution de son programme qu'il doit alors interpréter ou corriger.

Ce découplage extrême des entrées et des sorties engendre une interaction extrêmement pauvre entre la machine et son utilisateur. Pour rentre l'outil informatique utilisable par un public toujours plus grand, chercheurs et industriels ont essayé de réduire ce découplage entre les entrées et les sorties. Le premier problème fut de réduire le temps entre la soumission de commandes et la réponse de la machine. Pour que l'être humain ait l'impression d'interagir avec la machine, il faut que celle-ci "réagisse" dans une fourchette de temps compatible avec les interactions entre humains.

Il a donc par la suite été possible de soumettre soi-même son programme à la machine et de récupérer les résultats en quelques heures. L'interaction homme-machine ne débuta que lorsque l'utilisateur disposa d'entrées et de sortie plus localisées en disposant d'un clavier et d'un écran en face de lui et d'un système disposant d'un programme chargé de "converser" avec lui en temps réel (c'est-à-dire dans des temps compatibles avec les réactions et la réflexion humaine) : un *shell*. À l'exception notable de Sketchpad [111, 112] de Ivan Sutherland (1963) avec son stylo optique permettant de designer des objets sur un écran, les débuts de l'interaction Homme-Machine reposent essentiellement sur des interface textuelles. Même les systèmes les plus aboutis de l'époque comme NLS présenté par Douglas Engelbart lors de sa fameuse démonstration en 1968 [1], sont vus comme des machines complexes, inutilisables par des utilisateurs novices. L'introduction d'objets "directement" manipulables par l'utilisateur est une révolution. Puis l'introduction d'affichage bitmap a permis l'introduction du concept de "manipulation directe" [106] selon lequel les objets graphiques peuvent être attrapés et déplacés

1. http://sloan.stanford.edu/MouseSite/1968Demo.html

comme des objets réels offrant une correspondance plus naturelle pour l'interaction.

L'informatique personnelle a vu le jour il y a maintenant plus de 30 ans avec le système Star de Xerox et a conceptuellement peu évolué depuis lors. Inversement, la quantité d'information aussi bien créée que gérée par les utilisateurs de ces systèmes a augmenté de manière significative. Cependant, les moyens d'entrée et de sortie sont toujours à l'heure actuelle découplés dans les interfaces utilisateurs. Les interfaces tangibles (TUI pour Tangible User Interfaces [104]) peuvent être vues comme la prochaine évolution de l'interaction entre l'homme et la machine en proposant un regroupement des entrées et des sorties. En effet, ces interfaces proposent d'augmenter des objets physiques afin de proposer une correspondance directe entre manipulation dans le monde virtuel et manipulation dans l'environnement. Toutefois, ce type d'interfaces, même si elles sont séduisantes et performantes pour des utilisations bien définies perd l'attrait multi-usage de l'informatique grand public. En effet, le monde virtuel permet d'outrepasser les limites du monde physique et, par là même, d'augmenter les capacités de l'être humain. Il peut donc être parfois contre-productif de chercher à être aussi proche que possible de la métaphore physique. Par exemple, dans le cas de la métaphore du bureau qui constitue notre objet d'étude, il serait extrêmement réducteur de n'offrir que des manipulations directement suggérées par l'objet physique manipulé, la souris.

La suite de cette thèse porte sur des interfaces graphiques comprenant notamment un environnement de travail utilisant la métaphore du bureau et un ensemble de fenêtres pour présenter des informations. L'environnement de bureau et le gestionnaire de fenêtres sont souvent regroupés sous l'appellation "*Système*".

1.1 La Notion de Fenêtre

La fenêtre est pour beaucoup d'utilisateurs un simple élément graphique, rectangulaire, qui apparaît lorsque l'utilisateur ouvre un document ou démarre une application. Un utilisateur moyen déplace, iconifie, ferme des dizaines, voire des centaines de fenêtre au cours d'une session de travail. La fenêtre fait véritablement partie de la vie quotidienne de tout utilisateur de l'outil informatique. Cependant, la fenêtre est devenue un concept transparent. La plupart des utilisateurs confondent, par métonymie, la fenêtre et son contenu : documents, applications, pages internet, palettes d'outils... Toutefois, malgré son apparente simplicité la fenêtre est plus complexe qu'il n'y paraît.

La fenêtre apparaît pour la première fois en tant qu'objet informatique dans la thèse d'Alan Kay [68]. Elle est alors la matérialisation d'un ordinateur. À cette

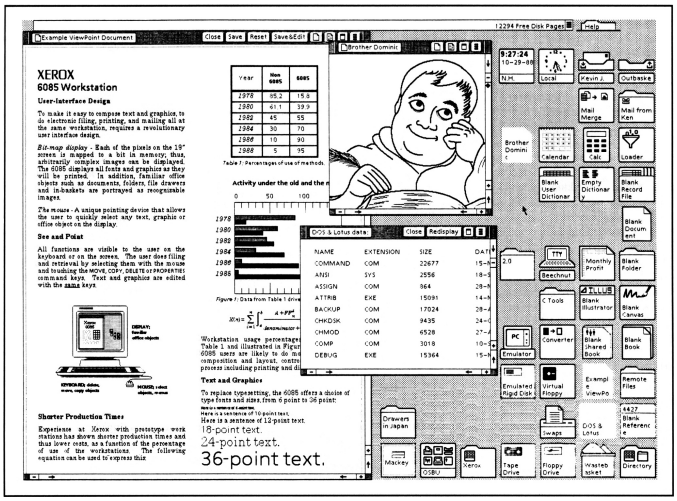


FIGURE 1.1 – *L'environnement graphique proposé par le Xerox Star (image tirée de [64]).*

époque la plupart des ordinateurs (ou plutôt les systèmes d'exploitation) ne permettaient d'exécuter qu'un seul programme à la fois. Même s'il existait certains systèmes plus avancés permettant d'exécuter simultanément différents programmes, la totalité de l'écran était allouée à l'application courante, les autres restent invisibles à l'utilisateur. De plus ces systèmes ne présentaient encore que des écrans en mode texte : tout ce qui était affiché n'était que des caractères à espacement fixe (comme les caractères tapés à la machine à écrire). Avec l'apparition d'écrans dit "bitmap" il fut possible d'afficher non plus des caractères mais des pixels, ouvrant la voie aux images et contenus plus fins. Le concept de fenêtre introduit par Alan Kay permettait alors de partager la ressource limitée qu'était l'espace d'affichage. En fait, pour son créateur, la fenêtre était à la fois un moyen de virtualiser les concepts d'ordinateur, de programme ou de tâche et de les matérialiser à l'écran pour les utilisateurs du système multitâches.

Le premier usage de la fenêtre dans un système commercial fut dans le Xerox Star en 1981 [64] (voir la Figure 1.1). Ce système, plus ou moins inspiré des recherches d'Alan Kay et de ses collègues au Xerox Parc, reprend l'idée d'utiliser la fenêtre comme matérialisation d'objets à l'écran. Le Xerox Star était une machine pensée en tant que système d'édition et de préparation de documents en vue de leur impression à l'aide des imprimantes laser. La fenêtre y perd alors son aspect d'ordinateur virtuel pour revêtir celui de document. Ceci est justifié par la cible principale du Xerox Star, à savoir les secrétaires de direction.

Le Xerox Star fut cependant un relatif échec commercial du fait principalement de son prix hors d'atteinte pour beaucoup d'entreprises. Ce fut alors Apple avec ses ordinateurs Lisa puis Macintosh (Figure 1.2) qui ont démocratisé les tech-

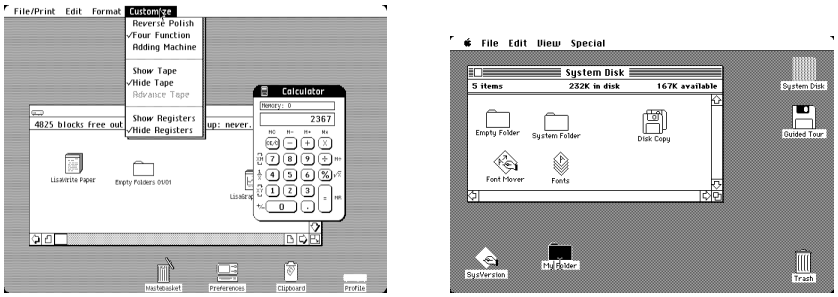


FIGURE 1.2 – *Apple Lisa et Mac OS.*

nologies d'écrans bitmap et d'interfaces utilisateurs graphiques. Ces machines sont fortement inspirées des travaux réalisés au laboratoire Palo Alto de Xerox et sont donc assez proches du Xerox Star. Cependant, l'introduction de nombreuses applications tierces développées pour le système Mac OS du Macintosh entraîne un nouveau changement dans l'idée que se fait l'utilisateur de la fenêtre : ici la fenêtre est synonyme d'application.

En résumé, si la fenêtre informatique en elle-même a peu évolué depuis son invention, la vue qu'en ont les utilisateurs a évolué au fil du développement des usages de l'informatique. Aujourd'hui, nous pouvons identifier trois grandes familles de fenêtres :

Application : c'est, avec la fenêtre en tant que document, la forme la plus répandue de fenêtre. C'est également la plus proche de ses origines en tant qu'élément fondamental du multitâche. Toutes les autres formes de fenêtres sont évidemment issues de celle d'application mais sont raffinées pour coller plus fidèlement à leur utilisation.

Document : le Xerox Star était un système visant les travailleurs de grandes entreprises et plus particulièrement l'édition ou les comptables. De ce fait les documents édités ou visualisés par l'utilisateur de ce système étaient centraux. Un document dans un tel environnement est indissociable de la fenêtre permettant de le visualiser et de le manipuler. C'est la fenêtre qui présente un document regroupant des contenus hétérogènes.

Objet d'interaction : cette dernière utilisation de la fenêtre se rapporte le plus souvent aux boîtes de dialogue et autres fenêtres modales. Ces fenêtres n'ont la plupart du temps qu'un contenu très standardisé informant l'utilisateur d'événements générés par le système, demandant confirmation à l'utilisateur ou lui posant des questions simples sur ce qu'il convient de faire.

A toutes ces utilisations de la fenêtre, liées à leur contenu, il convient d'ajouter l'aspect fonctionnel de la fenêtre. Du point de vue de l'utilisateur, elle permet la gestion de son espace d'affichage.

1.2 Gestion de fenêtres

Le composant du système permettant à l'utilisateur d'interagir avec les fenêtres et de les manipuler est appelé le gestionnaire de fenêtres. Nous suivons ici la terminologie de Myers [85] qui parle de "système de fenêtrage" pour les couches de bas niveau (rendu des fenêtres et capture des entrées) et utilise "gestionnaire de fenêtres" pour la partie qui se charge de l'interaction avec l'utilisateur : décoration des fenêtres, placement des fenêtres, opération de déplacement, iconification...

Le modèle de gestionnaire de fenêtres le plus répandu est celui dit des fenêtres recouvrantes. Dans ce modèle, les fenêtres sont disposées dans un espace en deux dimensions mais possèdent également une profondeur, c'est pourquoi certains qualifient parfois ce type d'espace de 2D 1/2 [23]. Les fenêtres peuvent donc se chevaucher, une fenêtre plus "haute" que les autres masque tout ou partie de celles qui se trouvent en dessous.

Cependant ce modèle de fenêtrage (ou de partage des ressources écran) a montré ses limites et surtout son incapacité à passer à l'échelle. On pourrait intituler ce mode de fonctionnement d'allocation manuelle d'espace d'affichage. Comme toute opération manuelle, si le nombre d'objets à allouer devient trop grand, l'utilisateur devient vite surchargé par la tâche d'allocation.

La recherche en IHM et l'industrie aussi bien que des développeurs indépendants enthousiastes ont tenté depuis plusieurs décennies de pallier à ces problèmes. Ces solutions vont de la conception de nouvelles interactions pour faciliter certains aspects de la gestion des fenêtres recouvrantes (typiquement le changement de fenêtre) à la conception de tout autre moyen de partager l'espace d'affichage.

1.2.1 Allocation de l'Espace d'Affichage

Un des problèmes majeurs de la gestion de fenêtres est l'allocation de l'espace d'affichage en deux dimensions, constitué par le (ou les) écran(s), aux différents contenus que l'utilisateur veut visualiser.

Alternatives au Chevauchement de Fenêtres

S'intéressant aux problèmes introduits par le recouvrement des fenêtres qui induit des zones invisibles pour l'utilisateur, une idée simple fait son apparition rapidement : ne plus permettre aux fenêtres de se chevaucher. Ce type de gestionnaires de fenêtres, par pavage de l'écran, a fait l'objet d'une étude dès 1986 [18] en le comparant au système de fenêtrage par chevauchement déjà considéré comme le

FIGURE 1.3 – *Elastic Windows*

fonctionnement prédominant. L'avantage de ce mode d'opération est de décharger l'utilisateur de nombreuses opérations de gestion des fenêtres par une gestion partiellement automatique de l'espace d'affichage mais au détriment du contrôle de l'utilisateur sur son espace d'affichage.

Afin de maximiser l'espace d'affichage utilisé, ce type de gestionnaire de fenêtres adopte souvent un comportement de remplissage : les fenêtres occupent le plus d'espace possible. Les premières versions du Xerox Star possédaient un gestionnaire de fenêtre qui utilisait un algorithme de placement des fenêtres conçu pour paver la surface de l'écran lorsque cela était possible.

Parmi les systèmes les plus populaires utilisés actuellement on peut citer Xmonad [109], Ion [2] et Ratpoison [3]. Ces gestionnaires de fenêtres ont été conçus pour une utilisation sur des ordinateurs portables et une interaction au clavier et permettent plus ou moins facilement et plus ou moins interactivement de modifier la façon dont le système dispose et redimensionne les fenêtres. Par exemple, Xmonad dispose d'un grand nombre d'agencements pensés pour des scénarios d'utilisation particuliers : Gimp, fenêtres de messagerie instantanée...

Elastic Windows [65] fait figure d'exception au sein des gestionnaires de fenêtres par pavage. Ce gestionnaire propose en effet, en plus des caractéristiques exposées ci-dessus, une gestion groupée et hiérarchique des fenêtres. Plusieurs fenêtres peuvent être regroupées au sein d'une nouvelle fenêtre, elle-même pouvant être groupée avec d'autres fenêtres. Ces regroupements créent une hiérarchie de

2. http://tuomov.iki.fi/software
3. http://www.nongnu.org/ratpoison

fenêtres sur laquelle il est possible d'interagir à tous les niveaux. L'allocation de l'espace d'affichage est gérée par le système sous forme de TreeMap [13] permettant d'utiliser la totalité de la surface d'affichage. Pour chaque groupe de fenêtres, ces dernières sont pavées dans leur fenêtre englobante de manière à occuper entièrement l'espace disponible. L'écran constitue la racine de cette hiérarchie (Figure 1.3).

Scheme Constraints Window Manager (SCWM) [8] est un autre exemple de gestionnaire de fenêtres qui automatise la gestion de l'espace d'affichage. SCWM propose d'augmenter un gestionnaire classique, permettant le chevauchement de fenêtres, avec des contraintes entre fenêtres définies par l'utilisateur et gérées par un algorithme de résolution de contraintes [7]. Cependant les interactions pour spécifier les contraintes sont peu naturelles et reposent majoritairement sur des boites de dialogues de style "formulaire".

La Tâche au Cœur de la Gestion de Fenêtres

Une autre piste largement explorée pour simplifier la gestion de fenêtres en réduisant le nombre d'entités à gérer simultanément tout en conservant la simplicité et la versatilité des gestionnaires de fenêtres classiques est d'introduire la notion de *tâche* ou d'*activité*. Cette notion, parfois définie de façon floue, permet de réduire le nombre de fenêtres en les groupant par utilisation et en n'affichant à un moment donné que les fenêtres utiles. En quelque sorte, la prise en compte des tâches ou des activités permet de diviser pour mieux régner.

L'incarnation la plus simple de ce principe se trouve dans les Espaces de Travail Virtuels et fut introduit par Rooms [51] en 1986 : Chaque pièce ("*room*") contient ses propres fenêtres. Une seule pièce est affichée à un instant donné, et l'on peut passer d'une pièce à l'autre par des portes. Dans certains systèmes, il est possible de définir des fenêtres qui sont affichées dans toutes les pièces, comme par exemple une horloge ou une application de messagerie. Cette solution, relativement simple, a rapidement été adoptée par nombre de gestionnaires de fenêtre destinés au système de fenêtrage X Window. Elle n'a fait son apparition que récemment en standard sous Mac OS X avec *Spaces* et est toujours absente (par défaut) de Microsoft Windows.

Une autre forme de groupement des fenêtres consiste à les imbriquer. Ainsi, dans d'anciennes versions de Windows, une application qui gére plusieurs documents consiste en un fenêtre représentant l'application, elle-même contenant une fenêtre pour chaque document. En pratique, les utilisateurs ont tendance à maximiser la fenêtre de l'application, reproduisant ainsi un modèle proche de Rooms mais contraint par le fait que seuls les documents d'une même application peuvent être affichés ensemble.

FIGURE 1.4 – *La Task Gallery [98]*

D'autres systèmes ont par la suite étendu et amélioré le concept de tâche. Parmi les plus marquants, se trouvent des gestionnaires de fenêtres comme Task Gallery [98] qui introduit la 3ème dimension (voir la Figure 1.4). D'autres implémentations raffinent le concept de tâche et en améliore la prise en considération dans le gestionnaire de fenêtres. Parmi eux se trouve WindowScape [116] qui propose de prendre des "instantanés" de l'organisation de l'espace d'affichage à un moment donné afin de pouvoir y revenir ultérieurement par un simple clic.

Pour sa part, Scalable Fabric [97] propose d'utiliser un système de "focus + contexte" pour gérer les tâches de l'utilisateur. Dans la partie centrale de l'écran, les fenêtres se comportent normalement. Mais lorsque l'utilisateur déplace une fenêtre vers un bord de l'écran, celle-ci réduit sa taille pour ne plus occuper qu'une petite partie de l'espace d'affichage. Lors de leur déplacement sur les bords de l'écran il est également possible d'approcher une fenêtre soit d'une autre fenêtre pour créer un groupe, soit d'un groupe pour l'y ajouter. Nommer un groupe de fenêtres crée une tâche. Cliquer sur une fenêtre ou un groupe de fenêtres a pour effet de replacer tous les éléments à leur place antérieure.

Enfin, Elastic Windows [65] permet de regrouper les fenêtres en tâches elles-mêmes peuvant être regroupées, créant ainsi une hérarchie de fenêtres. Il est possible d'appliquer des opérations à n'importe lequel de niveau hérarchique. Ces propriétés alliées à l'organisation des fenêtres par pavage 1.3 d'Elastic Windows permettent une gestion aisée des tâches. Par exemple, un programmeur pourrait organiser son espace en fonction des projets sur lesquels il travail, chaque projet se subdivisant en documentation et fichiers édités. Ainsi, le changement de projet deviens facile : paquer le projet courant puis dépaquer le projet suivant. De même lorsque l'utilisateur n'a pas besoin de la documentation, celle-ci peut rester paqué

libérant de l'espace d'affichage pour l'édition des fichiers.

Contourner le Chevauchement

La possibilité de chevauchement des fenêtres est un avantage puisqu'elle permet d'étendre virtuellement la surface d'affichage en rendant possible l'affichage de plus d'informations que ne le permettrait normalement l'écran. Cependant, elle entraîne une surcharge de manipulations pour l'utilisateur qui doit alors gérer ce recouvrement. Plusieurs tentatives ont été faites au cours du dernier quart de siècle pour résoudre le problème d'allocation par différentes techniques d'interaction semi-automatiques aidant l'utilisateur à gérer au mieux son espace d'affichage tout en permettant de garder cette possibilité de chevauchement.

Ishak et Feiner [62] (voir aussi [117]) proposent d'utiliser la transparence de façon contrôlée (uniquement sur des parties non importantes d'une fenêtre) pour visualiser le contenu, normalement caché de fenêtres recouvertes (partiellement ou complètement). Ils proposent également de pouvoir interagir avec le contenu ainsi découvert.

En améliorant une technique simple pour trouver l'emplacement idéal pour placer un rectangle, sans recouvrement avec d'autres rectangles déjà placés [15], Bell et Feiner [14] proposent une représentation de l'espace d'affichage permettant de placer les fenêtres, qu'elles soient créées ou déplacées, à un emplacement empêchant le recouvrement des fenêtres existantes. Cette technique permet de bénéficier de nombre des avantages des gestionnaires de fenêtres par pavage tout en conservant les avantages offerts par la possibilité de faire chevaucher les fenêtres. Dans le même ordre d'idée, Hutchings et Stasko [59] définissent deux nouvelles opérations sur une (des) fenêtre(s) afin de faire "grandir" celle(s)-ci le plus possible sans qu'elle ne recouvre les autres fenêtres. La première opération agrandit simplement la fenêtre courante jusqu'à ce que ses bords ne buttent sur d'autres fenêtres. La seconde, en plus d'agrandir la fenêtre courante, pousse également les autres fenêtres jusqu'à ce que l'ensemble des fenêtres occupe tout l'espace d'affichage disponible (l'écran) sans chevauchement.

Métisse [26], un système de fenêtrage X Window expérimental comportant un gestionnaire de fenêtres qui effectue son rendu en OpenGL, a exploré d'autres façons de réduire les chevauchements. Ainsi, Métisse propose à l'utilisateur des opérations de zoom et de rotation 3D pouvant être utilisées pour réduire le recouvrement entre fenêtres. Le zoom est différent du redimensionnement car il applique un facteur d'échelle uniforme à tout le contenu de la fenêtre, ce qui évite de changer son agencement. La rotation produit l'effet d'une porte que l'on ouvre, l'effet de perspective permettant de réduire sa largeur.

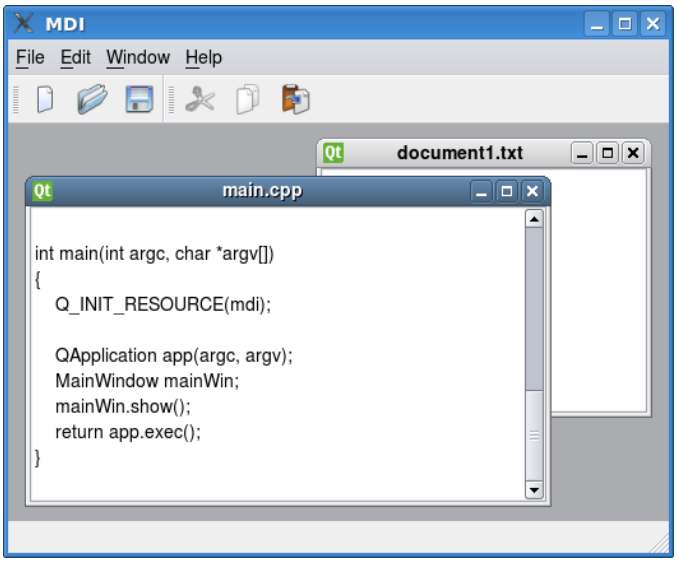


FIGURE 1.5 – *QT Inner windows*

Autres Solutions

Devant la faible évolution des systèmes de fenêtrage ou tout du moins la faible adoption des avancées de la recherche dans les systèmes les plus répandus, les concepteurs d'applications ont commencé à implémenter eux-mêmes de mini-gestionnaires de contenus.

Les bibliothèques de composants graphiques de nombre de boîtes à outils de construction de logiciel (QT ou Java/Swing par exemple) fournissent en standard des espaces d'affichage pouvant contenir leurs propres fenêtres. Ces fenêtres sont très souvent "dockables". Ceci signifie que, si l'utilisateur les approche du bord de la zone d'affichage, ces fenêtres se collent au bord, délaissant leur habillage et se comportant alors comme une boîte à outils. Ces mêmes bibliothèques proposent bien souvent la gestion plus ou moins avancée d'onglets qui sont un autre moyen de partager les ressources d'interaction entre différents contenus. Ce système d'onglets a d'ailleurs été utilisé en 2001 par Beaudouin-Lafon [12] comme une manière d'améliorer les gestionnaires de fenêtres par chevauchement.

Un autre exemple de cette mouvance sont les navigateurs Internet. Ces logiciels se sont vus forcés d'évoluer fortement depuis ces dix dernières années devant l'explosion de l'utilisation de l'Internet et l'apparition de nombreuses applications en ligne et du "cloud computing". Les utilisateurs passent de plus en plus de temps uniquement dans le navigateur qui devient peu à peu le nouveau "système". Or, les développeurs de ces navigateurs ont depuis longtemps compris les limitations des systèmes de fenêtrage classique. Depuis ses débuts, Internet est surtout utilisé

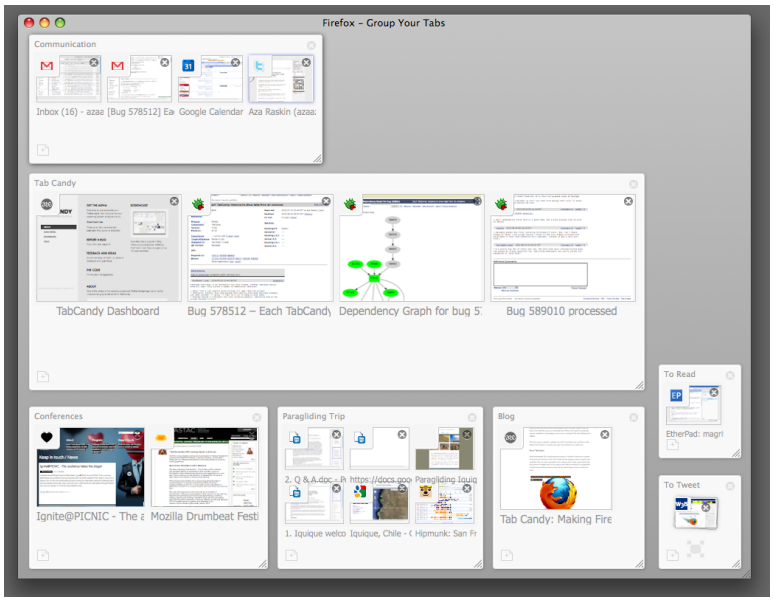


FIGURE 1.6 – *L'outil Panorama introduit dans Firefox 6 permet de gérer de façon efficace, pratique et élégante un grand nombre de pages Internet. (Image réalisée par Aza Raskin.)*

comme une ressource documentaire. De ce fait, une session de navigation sur Internet est le plus souvent une tâche d'exploration générant l'ouverture d'une multitude de pages Internet entraînant rapidement une surcharge des capacités de gestion de l'utilisateur muni des outils qui lui sont fournis par le système. Les concepteurs de navigateurs ont alors opté pour gérer l'ouverture des pages au sein même de leur application et ont adopté les onglets comme éléments de base, plus adaptés que les fenêtres recouvrantes pour l'exploration d'Internet : un onglet ouvert plus récemment est placé à droite d'un onglet ouvert il y a plus longtemps. Cette tendance s'accentue : À l'heure où les navigateurs Internet veulent devenir les nouveaux systèmes d'exploitation, ils continuent d'intégrer bien plus rapidement les éléments issus de la recherche que ne le font les systèmes traditionnels. Par exemple, le navigateur Firefox intègre un gestionnaire de pages qui propose les fonctions d'Exposé d'Apple et de bureau virtuel dans son outil Panorama [93] (Figure 1.6).

10/GUI[4] propose une approche différente de la gestion de fenêtres. Ce système prône l'utilisation d'une surface tactile multi-points à la place de la souris. Cette surface, aux dimensions de l'écran, permet une correspondance 1 :1 entre l'espace d'interaction et l'espace d'affichage. Au contraire des dispositifs mobile actuels, la surface tactile n'est pas confondue avec l'écran mais prends la place de la souris pour éviter les problèmes d'occlultation. 10/GUI cherche à simplifier la gestion des fenêtres en la limitant à une tâche en une seule dimension. Les fenêtres

4. http://www.10gui.com

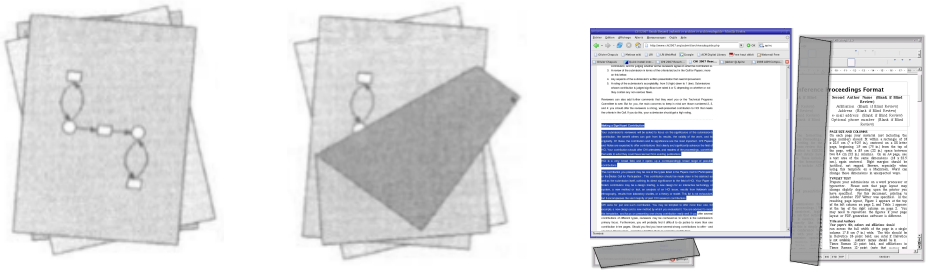
sont disposées linéairement, apparaissant à droite de l'écran et poussant les fenêtres précédentes vers la gauche, possiblement hors de vue. Les fenêtres occupent toutes la hauteur de l'écran et ne sont redimensionnables qu'en largeur. La puissance d'expression nécessaire à la gestion des fenêtres vient alors de l'utilisation de la technologie tactile multi-points. 10/GUI propose un vocabulaire de gestes dont la sémantique, c'est-à-dire la correspondance avec les commandes, dépend fonction du nombre de doigts utilisés. Les interactions avec un ou deux doigts sont destinées aux applications, alors que les interaction utilisant trois ou quatre doigts sont destinées au système. Il est ainsi possible de dezoomer pour avoir une vue plus large du ruban formé par les applications afin de trouver celle que l'on désire, réarranger les fenêtres ou tout simplement faire défiler le ruban.

1.2.2 Sélection et Changement de Fenêtre

La seconde opération essentielle d'un gestionnaire de fenêtres est la sélection de la fenêtre courante. Fondamentalement, cette opération se réduit à un problème de recherche. L'utilisateur à une idée de l'information qu'il recherche. Il doit retrouver la fenêtre qui contient cette information parmi toutes celles actuellement ouvertes. Cette sélection est typiquement effectuée de trois manières différentes :

- À l'aide de la souris : L'utilisateur connaît déjà la fenêtre hébergeant l'information recherchée et au moins une partie de cette fenêtre est actuellement visible. Selon le système et les préférences de l'utilisateur, deux variantes d'activation sont généralement disponibles : 1) Le système considère que la fenêtre active est celle sous le curseur de la souris ("focus follows mouse") ; 2) Le système considère que la fenêtre active est la dernière dans laquelle l'utilisateur a cliqué ("click to focus").
- À l'aide du clavier : Une combinaison de touches tapées sur le clavier (généralement `Alt-Tab`) permet de sélectionner successivement toutes les fenêtres. Là encore, plusieurs possibilités existent, allant d'une simple liste contenant le titre de chaque fenêtre à des miniature en temps réel des fenêtres.
- À l'aide d'objets tiers, comme la barre des tâches sous Windows ou tout autre système d'icônes.

Afin de faciliter l'opération de sélection de fenêtre à l'aide du dispositif de pointage et devant le problème posé par la possibilité de recouvrement total des fenêtres, Beaudouin-Lafon [12] propose de pousser plus avant la métaphore de la fenêtre en tant que document. Il définit deux nouvelles opérations. La première opération permet de "déranger" des piles de fenêtres comme il est possible de le faire avec une pile de feuilles de papier en inclinant plus ou moins les fenêtres de la pile (voir la Figure 1.7 (a)). Une fois cette opération appliquée, il est plus aisé pour l'utilisateur de sélectionner l'une des fenêtres se trouvant au-dessous. La



(a) (b) (c)

FIGURE 1.7 – *Aller plus loin avec la métaphore de la fenêtre en tant que document : deux nouvelles interaction pour facilité la gestion des piles de fenêtres. (a) "Déranger" la pile. (b) Plier les fenêtres de dessus. (c) Enroulement des fenêtres se trouvant devant une fenêtre lors d'une sélection. (Images tirées de [12] et de [27])*

seconde opération est similaire à celle effectuée lorsqu'un utilisateur recherche un document particulier dans une pile de feuilles : il replie partiellement chacune des feuilles jusqu'à trouver celle qui l'intéresse (voir la Figure 1.7 (b)). Cette dernière possibilité à ensuite été reprise et améliorée par l'interaction Fold'n'Drop [32] dans le cadre de l'utilisation du glisser-déposer. Le repliement des fenêtres est aussi utilisé dans la technique Rock-and-Roll [27] pour faciliter l'opération de copier-coller entre des fenêtres qui se recouvrent partiellement.

Une technique telle que `Alt-Tab` ne permet d'étudier qu'une seule fenêtre à la fois, tandis que la barre des tâches réduit chaque fenêtre à son simple titre. Devant ces problèmes, Mac OS X propose comme nouveau mécanisme de changement de fenêtre *Exposé* dans lequel le système réduit et dispose toutes les fenêtres de façon à les rendre toutes entièrement visibles (voir la Figure 1.8).

À l'instar de l'opération d'allocation, l'opération de sélection a elle aussi bénéficié d'une approche par tâche. En fait, les techniques d'allocation d'espace d'affichage comme Rooms, Window Scape ou Scalable Fabric, présentées dans la section 1.2.1, sont également considérées comme des techniques aidant la sélection de fenêtre. En effet, en plus d'affecter l'allocation de l'espace d'affichage, ces techniques impactent également la sélection d'une fenêtre puisque la sélection dépend alors des tâches auxquelles appartient la fenêtre recherchée. D'autres travaux se sont également penché sur la possibilité d'utiliser ce concept de plus haut niveau qu'est la tâche pour faciliter la sélection de fenêtres.

La barre des tâches, telle que celle utilisable sous Windows, possède un espace d'affichage limité. Lorsque trop de fenêtres sont ouvertes il devient impossible de toutes les représenter dans la barre, rendant cette technique quasiment inutilisable. En prenant cette limite en considération, Microsoft a proposé de grouper les fenêtres par application dans la barre des tâches. Par la suite GroupBar [89] améliore

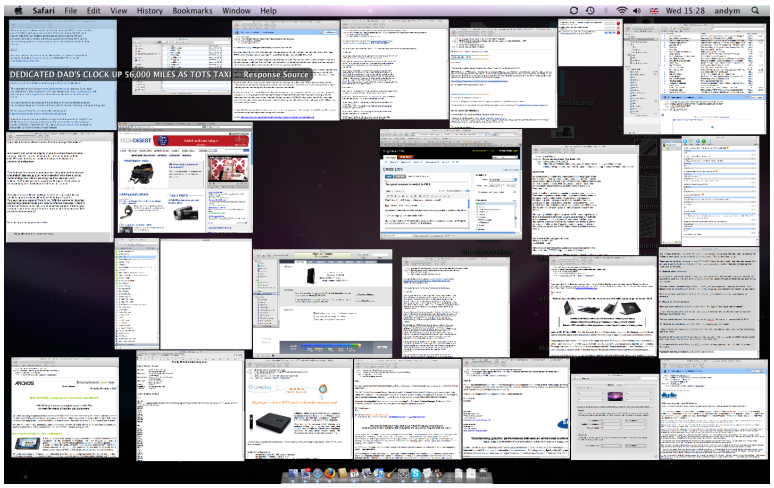


FIGURE 1.8 – *Exposé d'Apple*

ce concept en proposant un regroupement par application selon des critères définis par l'utilisateur, en général la tâche.

Les techniques de sélection par tâche présentées précédemment pourraient être qualifiées de manuelles. Elles requièrent de l'utilisateur qu'il indique par lui-même quelle fenêtre participe à quelle tâche. Des techniques comme Taskposé [16], RelAltTab[87] SWISH [88] ou UEMA [67] sont, pour leur part, des outils semi-automatiques. Elles essaient de créer des associations entre fenêtres d'après les actions de l'utilisateur. Par exemple, Taskposé utilise le passage d'une fenêtre `A` à une fenêtre `B` pour définir une relation entre `A` et `B`. Plus l'utilisateur passe de `A` à `B` (ou de `B` à `A`) plus cette relation devient forte. Lorsque l'utilisateur appelle Taskposé, le système crée alors, à l'aide de miniatures des fenêtres, une scène prenant en compte les liens entre les fenêtres : plus deux fenêtres ont une relation forte, plus elles sont proche l'une de l'autre dans la scène créée (voir la Figure 1.9).

1.3 Spatialité, Stabilité et Localité d'Interaction

Les êtres humains ont une facilité naturelle pour retrouver les objets dans le monde réel. Une large littérature existe à ce sujet en psychologie expérimentale (voir [20] pour une introduction). Cette faculté est également présente dans le cas d'environnements virtuels où des utilisateurs sont capables de retrouver des documents en fonction de leur emplacement des semaines après leur dernière visite [96]. L'organisation spatiale d'un environnement est donc important et doit rester stable pour qu'une personne soit capable de retrouver l'information qu'elle désire.

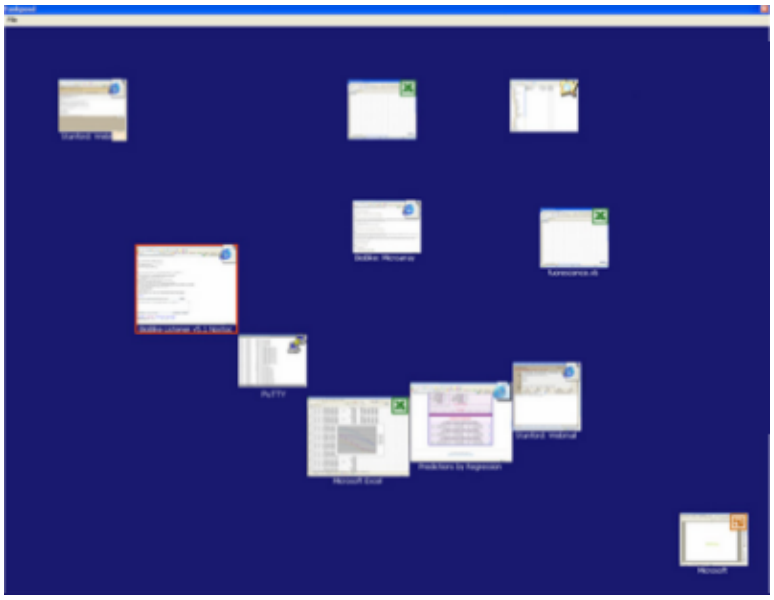


FIGURE 1.9 – *Taskposé : Les fenêtres possédant un lien fort (de nombreux passages de l'une à l'autre) deviennent de plus en plus proches. Des groupes de fenêtres émergent alors, pouvant indiquer que ces fenêtres participent à la même "tâche". Image tirée de [16].*

Les utilisateurs d'un système informatique effectuent souvent plusieurs tâches en parallèle (multi-tâches). Ils mènent de front un grand nombre de tâches qui s'entrecroisent, se chevauchent... Le changement de tâche et la reprise d'une tâche est toujours une opération coûteuse : remettre le système dans une configuration adéquate pour la prochaine session de travail sur la tâche, se remémorer les prochaines actions à effectuer sur la prochaine tâche [31]. La stabilité spatiale est un des aspects important de la gestion de fenêtre [115] dans ce contexte de tâche. Même pendant qu'il travaille à une tâche, l'utilisateur doit souvent changer de fenêtre pour accéder à différentes ressources, déplacer ou copier certains objets vers différentes destinations. Toutes ces opérations, nécessaires à l'accomplissement de la tâche courante, peuvent distraire l'utilisateur, lui imposant de nouveau changement de contexte (à la différence de la tâche). Souvent ces petites interruptions sont dues au fait que l'utilisateur doit déporter son attention sur une autre partie de l'interface afin d'effectuer des opérations secondaires.

1.4 Le problème

Lors de la création du concept de fenêtre et de leur apparition dans un système commercial tel que le Xerox Star, la puissance des machines et les usages limités

offerts par les systèmes informatiques limitaient le nombre de fenêtres ouvertes en pratique dans l'espace de travail. Cependant, les possibilités offertes par les systèmes informatiques se sont grandement diversifiées, la puissance des machines a considérablement augmentée. Qui plus est, la taille de l'affichage disponible a également connu une augmentation depuis l'introduction du concept de fenêtre. Dans leurs travaux, Hutchings et Stasko [60] observent, par exemple, que l'augmentation de l'espace d'affichage entraîne la disparition du comportement de "maximisation" des fenêtres, pourtant rependu dans les premiers temps des gestionnaires de fenêtres. Du fait de l'expansion de l'espace d'affichage, d'autres comportements ont également fait leur apparition comme la conservation des fenêtres comme mémos visuels afin de se souvenir d'une tâche restant à effectuer [60]. Enfin, l'espace d'affichage lui-même s'est modifié. Il n'est plus rare que cet espace d'affichage soit composé de plusieurs surfaces introduisant une complexité nouvelle dans la gestion de fenêtres [58]. La combinaison de ces changements a entraîné une augmentation du nombre de fenêtres ouvertes à un moment donné.

Parallèlement, les outils mis à la disposition de l'utilisateur pour gérer ses fenêtres n'ont que peu évolué. La plupart des environnements proposés sous le système Linux ont bien adopté un système d'espaces de travail multiples inspiré de Rooms [51] depuis quelques temps déjà, mais c'est l'une des rares avancées dans ce domaine avec les modes Exposé de Mac OS X.

Le résultat est une surcharge d'interaction pour l'utilisateur qui doit gérer de plus en plus d'informations à l'aide d'outils vieillissants, dont la majorité ont été pensés et conçus dans les premiers temps des interfaces fenêtrées, mais qui peinent de nos jours à épauler efficacement l'utilisateur.

Bien sûr, quelques systèmes industriels ou projets de recherche prometteurs cherchent à l'heure actuelle à résoudre les problèmes de gestion de fenêtres en regroupant les fenêtres, en utilisant des visualisations ou en intégrant dans leur fonctionnement des artefacts de plus haut niveaux comme les tâches. Ils ne sont pas sans défaut de notre point de vue. Beaucoup de ces techniques reposant sur les tâches ou proposant de regrouper les fenêtres requièrent de l'utilisateur des manipulations supplémentaires afin de pouvoir être utilisées. Les visualisations pour leur part arrangent les fenêtres selon leurs besoins avant de les présenter à l'utilisateur. Or, les êtres humains ont une mémoire spatiale développée dont ces interactions au mieux ne tirent pas parti, au pire, privent l'utilisateur d'une de ses capacités d'organisation en invalidant sa mémoire spatiale.

Finalement, si créer de nouvelles techniques d'interaction est important, les intégrer dans les systèmes existants est une tâche tout aussi importante. Il n'est pas rare de voir une fonctionnalité prometteuse devenir peu commode d'utilisation, car ses concepteurs n'ont pas réussi à l'intégrer de manière non intrusive dans l'écosystème de l'utilisateur. Les gestionnaires de fenêtres souffrent particulièrement de ce problème. De fait, les gestionnaires de fenêtres sont un artefact introduit par le

système. Ils offrent des services aux applications qui à leur tour offrent des services à l'utilisateur. La diversité de ces applications et le nombre des fonctionnalités offertes tendent à surcharger les canaux classiques d'interaction que sont la souris et le clavier, laissant peu de possibilités au gestionnaire de fenêtres afin d'offrir ses propres fonctionnalités à l'utilisateur sans interférer avec les applications. Ces constatations poussent trop souvent les concepteurs soit à complètement ignorer le problème de l'intégration soit à introduire leurs nouvelles techniques d'interaction dans les systèmes existants en utilisant des combinaisons de touches complexes ou en proposant une interaction à la souris nécessitant de l'utilisateur qu'il déporte son attention pour pouvoir déclencher ces techniques.

Les travaux que nous présentons dans la suite de cette thèse essaient de corriger les problèmes évoqués ci-dessus. Nous décrivons tout d'abord un ensemble de techniques d'interaction que nous avons conçu en réponse à ces problèmes. En particulier, elles tentent d'utiliser ces capacités innées de l'être humain que sont la cognition et la mémoire spatiales pour alléger la charge cognitive de l'utilisateur lors de son utilisation de systèmes de fenêtrage. Plus simplement, ces interactions prônent la stabilité spatiale de l'environnement afin de tirer parti de la mémoire spatiale des utilisateurs. De plus elles sont intégrées dans un système réel et utilisable sans que l'utilisateur ait à réaliser d'action compliquée, à changer de dispositif d'entrée ou à déporter son attention sur une autre partie de l'interface.

Le dernière partie de cette thèse s'intéresse à des fragments du contexte ou de la mémoire spatiale qu'il nous semblait important d'approfondir après avoir conçu des techniques d'interaction les utilisant : le pointage sur des cibles surgissantes ou animées dont la position a été mémorisé par l'utilisateur et une étude sur la perception de la profondeur en gestion des fenêtres. Enfin nous présentons des travaux préliminaires concernant l'utilisation du rythme comme méthode d'entrée. Nous voulions élargir et généraliser des techniques comme le double clic et les interactions temporisées présentées dans la première partie de cette thèse (voir la figure 2.2). Cette dimension d'interaction permet d'augmenter le pouvoir d'expression de l'utilisateur tout en ayant la possibilité de préserver la localité de l'interaction.

CHAPITRE 2

Techniques d'Interaction : Power Tools

2.1 Introduction

Dans ce chapitre nous présentons trois techniques d'interaction, ainsi que leur intégration dans un environnement de bureau réel et utilisable. Ces trois techniques forment la base de la présente thèse. Ce sont des interactions génériques qui ont été conçues dans l'optique d'être utilisées lors d'opérations de déplacement ou de copie d'éléments (texte, fichier, image...).

Les exemples caractéristiques d'opérations de déplacement et de copier sont le copier-coller, le couper-coller et le glisser-déposer. Ces opérations, appelées ci-après *copier-ou-déplacer*, sont des mécanismes de communication inter-applications importants. Elles définissent un protocole d'échange d'informations simple et standardisé entre les différentes applications, au niveau de l'interaction. Elles aident ainsi la créativité de l'utilisateur [107].

Il est rare, de nos jours, de concevoir un document à partir d'une page blanche. Une grande partie d'un document est souvent composé par l'utilisateur en réutilisant des parties de documents plus anciens, que ce soit des éléments de style ou du contenu. De plus, la plus petite tâche d'édition requiert, dans bien des cas, l'utilisation de plusieurs logiciels intégrant les contenus dans les documents maîtres. Les processus de création utilisent chaque application spécialisée dans une sous-activité spéciale. En ce sens, les applications sont très proches de ce que l'on pourrait qualifier "d'outil" informatique. En dehors de cette aide à la créativité, ces opérations sont simplement indispensables dans l'utilisation quotidienne de l'outil informatique. On utilise typiquement une opération de copier-coller pour une simple recherche d'un terme inconnu dans un dictionnaire en ligne ou la recherche sur le web d'informations supplémentaires sur un ouvrage en copiant son titre . Ces opérations qui paraissent si simple seraient très fastidieuses sans de tel mécanismes.

Une preuve supplémentaire de la popularité et de l'utilité de ces opérations est les différentes tentatives d'adaptation de ces opérations à des contextes multi-systèmes avec par exemple le Pick-and-Drop de Rekimoto [95] et le presse-papiers ubiquitaire de Miller [81]. La transmission d'informations entre applications s'exécutant sur des systèmes différents est encore à l'heure actuelle une opération complexe qui requiert souvent de l'utilisateur, dans le meilleurs des cas, des paramétrages complexes et, dans le pire des cas, une compatibilité entre les systèmes souvent difficile à atteindre.

Le but premier des interactions présentées dans ce chapitre est de faciliter l'usage des opérations de *copier-ou-déplacer* dans un environnement de bureau. Pour cela, elles prennent en considération le fait qu'elles sont souvent secondaires pour l'utilisateur. Il est donc important de ne pas bouleverser l'environnement de celui-ci juste pour l'utilisation de ces outils. Elles conservent donc, lorsque c'est possible, la disposition des éléments à l'écran et ne forcent pas l'utilisateur à déporter son attention vers d'autres emplacements : elles conservent le flot d'interaction de l'utilisateur.

La première interaction proposée, *TimeShift*, s'intéresse aux historiques d'interaction. Elle s'inspire des historiques de copier-coller (presse-papiers) qui existent déjà tout en généralisant leur fonctionnement à d'autres opérations de *copier-ou-déplacer*. De plus, *TimeShift* a bénéficié d'une attention toute particulière concernant son interaction. Cette dernière ne force pas l'utilisateur à déporter son attention vers une autre partie de l'écran. Pour cela, l'appel des historiques se fait par des interactions semblables aux opérations classiques, augmentées à l'aide de la dimension temporelle.

Les deux interactions suivantes, *DeskPop* et *StackLeafing*, se concentrent sur l'épineux problème de la navigation entre les fenêtres au cours d'une interaction de glisser-déposer. Le glisser-déposer est une technique intéressante car elle renforce l'impression d'interagir directement avec les objets manipulés. Cependant, elle entraîne de sérieuses complications pour l'interaction avec le reste du système pendant son déroulement. En effet, lors de l'utilisation du glisser-déposer, l'utilisateur "tient" l'objet avec lequel il interagit en maintenant la pression sur le bouton de la souris. Cette pression constante et prolongée du bouton empêche la plupart des interactions naturelles à la souris avec les éléments de l'interface, par exemple pour déplacer des fenêtres.

DeskPop, présenté dans la section 2.4, est une technique permettant de visualiser entièrement l'espace du bureau (l'élément à l'arrière plan de l'écran dans la métaphore du bureau) afin de pouvoir interagir avec son contenu. Cette technique d'interaction répond aux mêmes motivations que des techniques chronologiquement antérieures telles que *Show Desktop* utilisé dans des environnements de bureau classiques tel que Microsoft Windows et un des mode de l'interaction Exposé de Mac OS X.

StackLeafing (voir section 2.5) est une technique servant à parcourir les fenêtres pour pouvoir voir leurs contenus. Elle s'apparente donc à des interactions comme `Alt-Tab` et toute autre technique permettant le changement de fenêtres. Ce qui la différencie est à la fois sa capacité à conserver la disposition dans le plan des différentes fenêtres en ne modifiant que leur empilement et sa capacité à grouper les fenêtres en couches pour pouvoir parcourir plus rapidement l'intégralité des contenus. *StackLeafing* incorpore également *DeskPop* pour afficher le bureau comme une couche à part entière.

StackLeafing et *DeskPop* sont intégrées dans l'environnement à l'aide d'un *Trailing Widget* (Figure 2.8). L'utilisation de ce moyen de déclenchement nous permet de disposer d'une interaction exclusivement à la souris sans avoir besoin d'interaction complexe.

Une vidéo illustrant les techniques présentées dans ce chapitre est disponible à l'url suivante `http://insitu.lri.fr/metisse/videos/powertools.mov`.

2.2 Interaction avec des Historiques de Sélection, de Déplacement et de Copie

Bien que les services offerts par les gestionnaires de presse-papiers semblent utiles et appréciés, force est de constater qu'ils sont assez peu utilisés en pratique [27]. Nous pensons que ceci vient du fait que ces outils obligent l'utilisateur à interrompre son flot d'interaction pour se servir du gestionnaire de presse-papiers. De plus étendre le comportement de la sélection de texte à des sélections multiples [82] entraîne également le besoin de ne plus gérer uniquement le cas simple de la dernière sélection, mais d'une liste d'objets potentiellement intéressants. Un exemple typique d'interaction avec un gestionnaire de presse-papiers peut être décrit par les actions suivantes :

1. Ouvrir la liste d'historique. Ceci requiert généralement de passer du clavier à la souris ou bien d'utiliser un raccourci clavier complexe, des actions déjà coûteuses.
2. Sélectionner l'élément qui nous intéresse à l'aide du clavier ou de la souris.
3. Retourner à l'emplacement de l'édition en cours avant de pouvoir se servir du gestionnaire de presse-papiers.
4. Enfin coller l'élément voulu à son emplacement en se servant d'une des techniques traditionnelles de copier (`Ctrl-V`, menu contextuel, clic sur le bouton du milieu de la souris).

L'interaction nécessaire à l'ouverture de la liste contenant l'historique du presse-papiers est l'une des parties problématiques de cet enchaînement d'actions. Nous pensons que c'est à ce niveau que les implémentations actuelles des gestionnaires de presse-papiers sont faibles. Les outils existants utilisent souvent des raccourcis clavier complexes à plus de deux modificateurs (`Shift+Ctrl+V` par exemple). Une autre possibilité fréquemment employée est l'utilisation d'une icône dans un coin de l'écran qui force l'utilisateur non seulement passer du clavier à la souris, mais nécessite également un déplacement conséquent de la souris. De plus ce type d'interaction oblige l'utilisateur à déporter son regard et son attention. Compte tenu des remarques précédentes, cette interaction doit être repensée pour éviter de perturber le travail de l'utilisateur. C'est cette volonté de conserver le flot d'interaction de l'utilisateur qui nous a poussé à revoir l'interaction avec les gestionnaires de presse-papiers. En particulier, l'interaction proposée pour accéder à nos historiques répond au priorité suivantes :

1. Réduire la navigation au maximum afin de permettre à l'utilisateur de conserver son attention sur son travail ;
2. Réduire le nombre d'interactions nécessaires en regroupant la sélection et l'insertion de l'élément ;
3. Garder une certaine consistance de l'interaction pour les opérations au clavier ou à la souris.

De plus, l'opération de copier-coller n'est pas la seule permettant le transfert d'informations entre application. Nous avons choisi d'étendre l'historique à d'autres opérations : le glisser-déposer et la sélection primaire de l'environnement X Window [1]. Ainsi, nous avons développé un outil permettant de garder une trace de chaque opération de copie, de sélection et de glisser-déposer (annulé ou non). Ces opérations sont regroupées en trois historiques distincts : de sélection, de copier-coller et de glisser déposer, accessibles au travers de trois menus déroulant séparés (voir la Figure 2.1).

Afin de respecter les contraintes d'interaction évoquées ci-dessus, nous avons utilisé le temps, une dimension encore sous-utilisée en interaction homme-machine. À l'instar d'Hinkley et collègues [53], nous avons utilisé un déclenchement contrôlé par un délai ("dwell") pour intégrer nos historiques en augmentant les interactions classiques auxquelles ils correspondent. En pratique, un délai est utilisé dans notre implémentation pour différencier les interactions classiques à la souris ou au clavier qui doivent être transmises aux applications et les commandes spéciales destinées au système de fenêtrage.

Cette méthode possède plusieurs avantages. Sur le plan technique, elle permet d'augmenter le pouvoir d'expression de séquences d'interaction classiques en

1. Avec X Window la dernière sélection (de texte) est copiée implicitement et peut être collée par un clic avec le bouton du milieu de la souris.

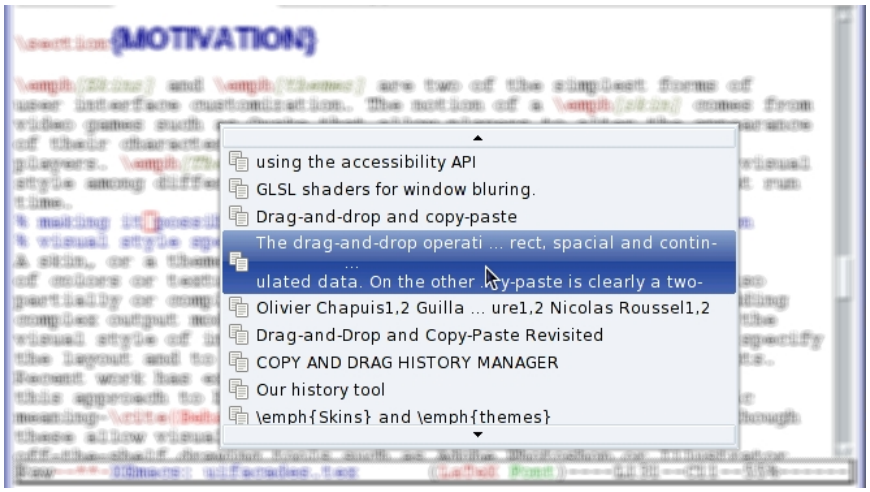


FIGURE 2.1 – *Un exemple de menu de sélection d'un élément d'historique.*

ayant un impact minimal sur les mécanismes de gestion des événements. Sur le plan de l'interaction, elle permet une séparation claire entre les interactions destinées aux applications et celles destinées au gestionnaire de fenêtres tout en tirant avantage des habitudes de l'utilisateur en augmentant les interactions classiques (contrainte 3). De plus, elle ne force l'utilisateur ni à changer de dispositif d'entrée ni à déporter son attention vers une autre partie de l'interface (contrainte 1). La machine à états présentée à la Figure 2.2 illustre ce mécanisme dans le cas d'une interaction à la souris.

Le menu contenant l'historique des éléments copiés peut être appelé par l'utilisateur au moyen d'un `Ctrl-V` long (c'est-à-dire en gardant ces touches appuyés pendant plus de 250 *ms*). Les utilisateurs peuvent alors sélectionner les éléments précédents de manière circulaire par des appuis successifs sur la touche `V` tout en conservant la touche `Ctrl` appuyée (de façon similaire à l'interaction bien connue permettant de naviguer entre les fenêtres : `Alt-Tab`). On peut également parcourir cet historique à l'aide de la souris, des touches directionnelles, ou des touches `G` et `F`, choisies pour leurs proximité avec la touche `V`. Un appui sur la touche `C` (ou `Esc`) annule le menu, alors que relâcher la touche `Ctrl` déclenche l'insertion du dernier élément sélectionné. Quelques commandes spécifiques à chaque élément sont disponibles par l'appui sur de touches du clavier ou par un sous menu accessible par un clic droit de la souris. La touche `R` permet de supprimer l'élément de l'historique, la touche `U` place l'élément au début de la liste. Enfin, la touche `W` change la sémantique de l'opération : au lieu d'insérer l'élément, le système indique la fenêtre source de cet élément (la fenêtre d'où l'élément a été collé) en la plaçant au-dessus des autres.

Un appui long sur le bouton du milieu de la souris appelle le menu affichant l'historique des éléments sélectionnés (sélection primaire X Window). Comme

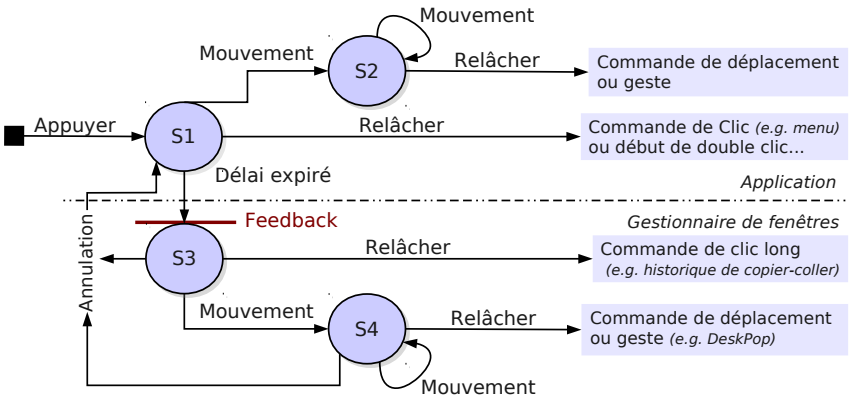


FIGURE 2.2 – *Les interactions temporisées permettent de faire la distinction entre le niveau d'interaction applicatif et le niveau gestion de fenêtres. Un feedback est donné à l'utilisateur au moment du passage de la barrière entre ces deux mondes.*

pour `Ctrl-V` nous prolongeons l'interaction classique en utilisant la dimension temporelle. L'utilisateur peut alors naviguer dans ce menu à l'aide de la souris et déclencher l'insertion d'un élément en cliquant dessus. Il est également possible de se servir de l'interaction de glisser-déposer en "arrachant" un élément du menu.

Un *clic* long (et non pas un appui long) sur le bouton droit de la souris appelle le menu contenant les éléments précédemment déplacés à la souris. Ce menu se comporte de façon similaire à celui des sélections à l'exception qu'un clic sur une élément commence une action de glisser-déposé libre : à la différence de l'interaction classique, l'utilisateur n'a pas besoin de garder le bouton de la souris appuyé lors du déplacement, mais doit alors cliquer pour indiquer la libération de l'objet. Un aspect intéressant de ce mécanisme d'historique des glisser-déposer est qu'il permet de reproduire partiellement l'interaction *Boomerang* [70] en permettant de reprendre plusieurs opérations de glisser-déposer interrompues : déplacer les objets de seulement quelques pixels, puis le relâcher pour annuler le déplacement, est suffisant pour faire entrer cette action dans l'historique. L'utilisateur peut alors continuer à ajouter des éléments de la même manière ou naviguer vers son point d'insertion et reprendre les objets depuis le menu d'historique pour les insérer à leur destination. De plus, notre système propose uns solution encore plus simple en se rappelant de toutes les sélections. Pour insérer les éléments dans l'historique, il n'est même pas nécessaire de les déplacer "artificiellement", leur sélection suffit.

Dans ce dernier cas nous ne prolongeons pas une action standard car celle-ci n'existe pas. L'action typique associée à un clic droit est l'apparition d'un menu contextuel. Le bouton droit est aussi utilisé dans certaines applications (e.g., Firefox) pour entrer des commandes via des gestes. Dans ce cas un clic droit appelle un menu contextuel, mais si l'on ne relâche pas le bouton on peut effectuer un geste

pour déclencher une commande de l'application courante (voir la partie haute de la Figure 2.2). C'est dans ce contexte que nous utilisons la dimension temporelle pour augmenter le pouvoir d'expression de l'utilisateur avec le système (et pas seulement l'application courante).

L'appui long sur le bouton droit indique que l'utilisateur veut agir au niveau du système et non pas de l'application courant. Comme décrit dans la Figure 2.2, après un feedback (l'apparition d'un petit carré rouge semi-transparent sous le curseur), l'utilisateur peut soit relâcher le bouton pour obtenir un menu (dans notre cas l'historique des glisser-déposer) ou bien faire un geste pour passer une commande au gestionnaire de fenêtre comme, par exemple, basculer vers le client de courriel, le navigateur web, ou déclencher les techniques *DeskPop* et *StackLeafing* de ce chapitre.

Afin de trouver des valeurs de délais raisonnables pour nos interactions, nous avons analysé les séquences de pression-relâchement sur les boutons du milieu et de droite de la souris. Les séquences analysées sont issues d'utilisateurs du système d'exploitation Linux lors de leur utilisation quotidienne de l'ordinateur [25]. Les enregistrements analysés comportent 32.571 clics avec le bouton du milieu de la souris et 65.980 clics avec le bouton droit de la souris (parmi 1.473.029 clics). Nous avons considéré le 95%-quantile de temps entre la pression sur le bouton et l'action débutant un glisser (un déplacement de 4 pixels) ou le relâchement du bouton. Les temps concernant les clics sur le bouton du milieu sont très réguliers entre les utilisateurs : une moyenne et une médiane de 22 *ms* avec une déviation standard de 44 *ms*. Cela confirme que le délai de 250 *ms* utilisé est adéquat. Au contraire, pour les clics sur le bouton droit de la souris, les valeurs enregistrées sont plus largement distribuées : une médiane de 222 *ms* et une moyenne de 417 *ms* avec une déviation standard de 310 *ms*. Cela nous a amenés à utiliser une valeur par défaut de 500 *ms* et montre que cette valeur devrait être paramétrable.

2.3 Comparaison de TimeShit à d'autres mécanisme d'historiques

Une étude *KLM-GOMS* [22] a été réalisée afin de donner une idée des forces et des faiblesses de notre implémentation des historiques de copier-coller et de les comparer à d'autres solutions existantes : le gestionnaire de copier-coller intégré à la suite bureautique de Microsoft (appelé *MS-Office-Clip* ci-après) et *Klipper*, le gestionnaire de copier-coller de l'environnement de bureau *KDE* (X Window).

Les valeurs utilisées dans cette analyse sont celles préconisées par David Kieras [69]. Ces valeurs sont reprises dans le tableau 2.1. Pour ce qui concerne l'action

KLM	Signification	Temps
M	Préparation mentale	1.20 *s*
H	Temps de passage entre souris et clavier	0.40 *s*
P	Temps de pointage	1.10 *s*
B	Pression ou relâchement d'un bouton de la souris	0.10 *s*
K	Temps pour taper une touche du clavier	0.20 *s*
$W(xx)$	Attente	xx *s*

TABLE 2.1 – *Valeurs des différentes opérations utilisées dans l'étude KLM-GOMS des historiques.*

K (entrée au clavier), il a été choisi pour cette étude la valeur de 200 *ms* qui correspond à celle décrite comme le temps moyen pour un utilisateur expérimenté (55 mots par minute). Nous avons choisi cette valeur afin de ne pas avantager les interactions uniquement au clavier (la valeur pour un utilisateur expert est de 120 *ms*) mais en considérant que l'usage d'un gestionnaire de copier-coller relève déjà d'une utilisation avancée, ce qui exclut la valeur pour une personne sans entraînement à la saisie clavier (280 *ms*).

Cette comparaison quantitative des performances des outils d'historique de copier-coller couramment utilisés sur 2 des 3 systèmes d'exploitation majeurs (Microsoft Windows et Linux) se propose d'étudier le cas d'un utilisateur voulant sélectionner l'avant dernier élément copié (le second de l'historique).

La première implémentation étudiée est celle incluse par la suite Office de Microsoft [2]. Cet outil est proposé conjointement à la suite Microsoft Office sur les systèmes d'exploitation Windows.

Plusieurs aspects de cet outil posent problème dans le cadre de la thèse exposée dans ce document :

- Tout d'abord, l'outil ne peut être utilisé qu'à l'aide de la souris. L'utilisateur doit de ce fait constamment changer de dispositif d'entrée afin d'effectuer ses opérations de copier-coller ;
- La liste affichant le contenu de l'historique de copier-coller se trouve sur le côté du document, obligeant l'utilisateur à déporter son attention hors du document pour pouvoir effectuer l'opération d'édition. Dans le pire des cas, l'utilisateur doit faire d'incessants aller-retours entre MS-Office-Clip et son document afin d'effectuer de multiples opérations de collage (ce qui, en définitive, est l'intérêt principal d'un tel outil).

2. http://office.microsoft.com/en-us/word-help/copy-and-paste-multiple-items-by-using-the-office-clipboard-HA010163602.aspx

KLM	Temps	Explication
H	0.40 s	L'utilisateur saisie la souris.
$MPBB$	2.50 s	L'utilisateur pointe et clic sur l'icône du gestionnaire afin d'en faire apparaître la fenêtre.
$MPBB$	2.50 s	L'utilisateur sélectionne l'élément désiré dans la liste proposé par le gestionnaire.
$MPBB$	2.50 s	L'utilisateur clic sur le bouton de fermeture du gestionnaire afin de le faire disparaître.
H	0.40 s	L'utilisateur pose de nouveau ses mains sur le clavier afin de continuer sa saisie.
Total	8.30 s	
	5.80 s	Sans la fermeture.
	3.30 s	Sans l'ouverture et sans la fermeture.

TABLE 2.2 – *Analyse KLM-GOMS pour le gestionnaire de copier-coller* MS-Office. *L'utilisateur selectionne la deuxième entrée de l'historique*

Cette implémentation possède d'autres problèmes qui peuvent impacter son utilité ou tout du moins son utilisabilité, mais qui ne sont pas relatifs à la thèse présentée ici :

- Cet outil est étroitement lié à la suite MS-Office. Lorsque l'un des outils de la suite est en cours d'exécution, le Clipboard enregistre bien dans son historique les opérations de copie effectuées hors des programmes de la suite, mais il n'est possible d'insérer des éléments que dans un des composants de cette suite logicielle ;
- Le contenu de l'historique conservé par cet outil est limité à 24 éléments.

Ces dernières limitations réserveraient l'utilisation d'un tel outil pour des tâches ponctuelles de copies multiples : l'utilisateur veut transférer le contenu d'un ou plusieurs documents dans un autre document. Il lance donc MS-Office-Clip et effectue toutes ses opérations de copie. Il va ensuite dans le document "maître" puis y insère les différents éléments dans l'ordre souhaité et aux emplacements désirés. Il ferme alors l'outil d'historique et continue d'éditer son document. Si cet outil rend ce cas d'utilisation plus confortable que des aller-retours entre tous les documents en jeu, il ne propose pas vraiment un aspect "historique" au sens entendu dans cette thèse.

L'étude KLM-GOMS de cet outil est présentée dans le tableau 2.2. Compte tenu des différentes méthodes d'activation de cet outil, plusieurs cas sont présentés dans ce tableau.

Le second outil d'historique de copier-coller étudié est nommé *Klipper*[3]. Il

3. http://docs.kde.org/stable/en/kdebase-workspace/klipper/index.

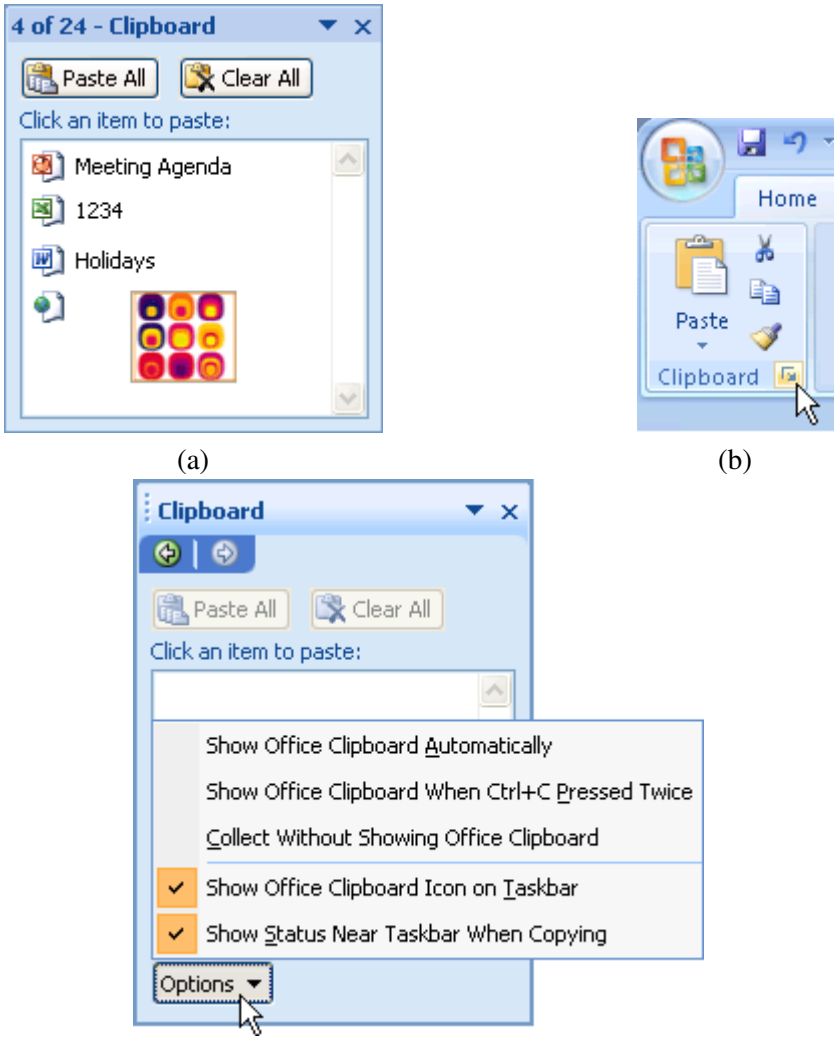


FIGURE 2.3 – *Le clipboard de Microsoft Office. (a) Liste d'historique affichant le contenu du presse-papier. (b) Ouvrir Microsoft Office Clipboard. (c) Configuration de l'affichage automatique de Microsoft Office Clipboard.*

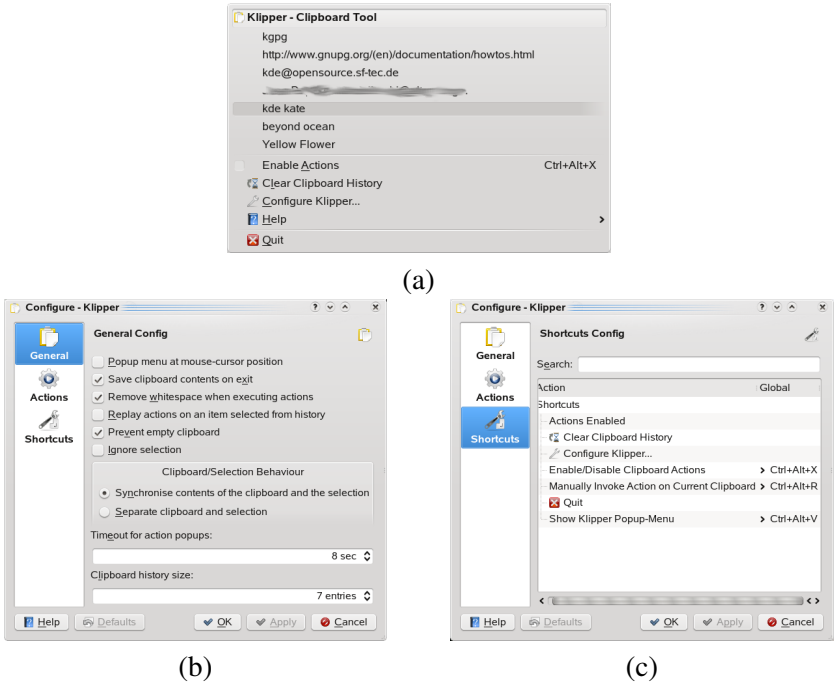


FIGURE 2.4 – *Klipper*

s'agit d'un instrument intégré à *KDE*[4], l'un des deux principaux environnements de bureau disponibles pour le système X Window. *Glipper*[5], une alternative plus récente, existe également pour *GNOME*, un autre environnement de bureau populaire pour X Window. Le fonctionnement de base de Glipper est très similaire à Klipper, même si le premier ne propose pas toutes les options de ce dernier.

Klipper peut être utilisé indifféremment à l'aide du clavier (voir table 2.3) ou

html

4. http://www.kde.org
5. https://launchpad.net/glipper

KML	Temps	Explication
$MKKK$	1.80 s	L'utilisateur appuis sur les touches `Ctrl+Atl+V` pour activer le menu de Klipper.
$MKKK$	1.80 s	Il sélectionne le second élément du menu et tape *Enter*.
MKK	1.60 s	Il appuis sur les touches `Crl+V` pour effectivement insérer l'élément sélectionné.
Total	5.20 s	

TABLE 2.3 – *Analyse* KML-GOMS *de l'interaction avec le gestionnaire de copier-coller* Klipper *à l'aide du clavier.*

KLM	Temps	Explication
H	0.40 *s*	L'utilisateur saisie la souris.
$MPBB$	2.50 *s*	L'utilisateur pointe et clic sur l'icône de Klipper afin d'en faire apparaître le menu.
$MPBB$	2.50 *s*	L'utilisateur sélectionne l'élément désiré dans la liste proposé par Klipper.
H	0.40 *s*	L'utilisateur pose de nouveau ses mains sur le clavier afin de continuer sa saisie.
Total	5.80 *s*	

TABLE 2.4 – *Analyse* KLM-GOMS *de l'interaction avec le gestionnaire de copier-coller* Klipper *à l'aide de la souris.*

KLM	Temps	Explication
MKK	1.60 *s*	L'utilisateur presse la combinaison de touches `Ctrl-V` : raccourcis clavier habituellement associer au "collage" du dernier élément "copié".
$W(0.25)$	0.25 *s*	L'utilisateur attends 250 *ms* que le système ne détecte qu'il désire appeler l'historique des copier-coller.
$MKKK$	1.80 *s*	L'utilisateur tape deux fois la touche `V` afin de sélectionner le second élément de la liste, puis il relâche la touche `Ctrl` pour indiquer la fin de l'interaction avec l'historique et la demande d'insertion de l'élément sélectionné.
Total	3.65 *s*	

TABLE 2.5 – *Analyse* KLM-GOMS *de l'interaction avec l'historique de copier-coller de* TimeShift.

de la souris (voir table 2.4). Par défaut, le menu affichant les éléments contenus dans l'historique de copie est, comme pour MS-Office-Clip, affiché à un emplacement éloigné du point d'intérêt (le point d'insertion dans le document en cours d'édition). Il existe depuis peu une option permettant de faire apparaître ce menu au niveau du curseur de la souris. Cependant, cet emplacement n'est pas forcément judicieux, le curseur pouvant se trouver (et même se trouvant souvent) à l'écart du point d'insertion. De plus, les raccourcis clavier utilisés, s'ils tentent d'être cohérents avec ceux utilisés par les opérations classiques (`Ctrl-V` devient `Ctrl-Alt-V`) en utilisant les mécanismes habituels pour résoudre ce genre de problème d'activation (ajouter un modificateur tel que `Ctrl`, `Alt` ou `Shift`) rendent l'interaction avec cet outil difficile. Il est possible de configurer le raccourci clavier utilisé pour invoquer le menu de sélection, mais le problème reste le même puisqu'il n'est pas possible que d'utiliser autre chose que des raccourcis classiques à base d'une touches du clavier, éventuellement avec des modificateurs.

Enfin, le dernier outil étudié est celui présenté dans ce chapitre : *TimeShift*. Son analyse KLM-GOMS est présentée par le tableau 2.5.

Au final, *TimeShift* requiert 3.65 *s* dans un cas d'utilisation simple de sélection de l'avant dernier élément copié. Seul MS-Office Clipboard – dans le cas particulier où il ne requiert ni ouverture ni fermeture de la barre latérale – propose une interaction à priori plus rapide que *TimeShift* en nécessitant 3.30 *s* pour effectuer la même opération. Cependant, plusieurs remarques s'imposent. Tout d'abord, cet outil est limité à un usage dans la suite MS-Office. Ensuite, dans le cas probable d'insertions multiples, l'utilisateur est forcé de faire des aller-retour entre le document et le Clipboard. Ces restrictions atténuent les gains de 10% enregistrés par cette interaction par rapport à *TimeShift*. Enfin, les performances des autres méthodes de déclenchement de MS-Office Clipboard sont elles en retrait par rapport à *TimeShift* (5.80 *s* et 8.30 *s* soit respectivement 60% et 127% plus lent). Finalement, *TimeShift* est plus rapide que Klipper quelque soit l'interaction utilisée (37% plus rapide pour l'interaction à la souris et 30% pour une interaction au clavier).

Ces analyses de l'interaction avec différent gestionnaires de presse-papier à un niveau très bas montrent que l'interaction offerte par l'outil *TimeShift* semble proposer, en plus d'une interaction plus centrée sur le point d'attention, des performances accrues par rapport aux outils classiques de gestion d'historiques de copier-coller. Ces résultats sont cohérents avec notre expérience de l'utilisation de *TimeShift*, et devraient être confirmés par des études plus approfondies.

2.4 DeskPop

Dans la métaphore du bureau, où les fenêtres jouent le rôle de documents, le bureau, c'est-à-dire la surface sur laquelle sont posés les documents et les outils,

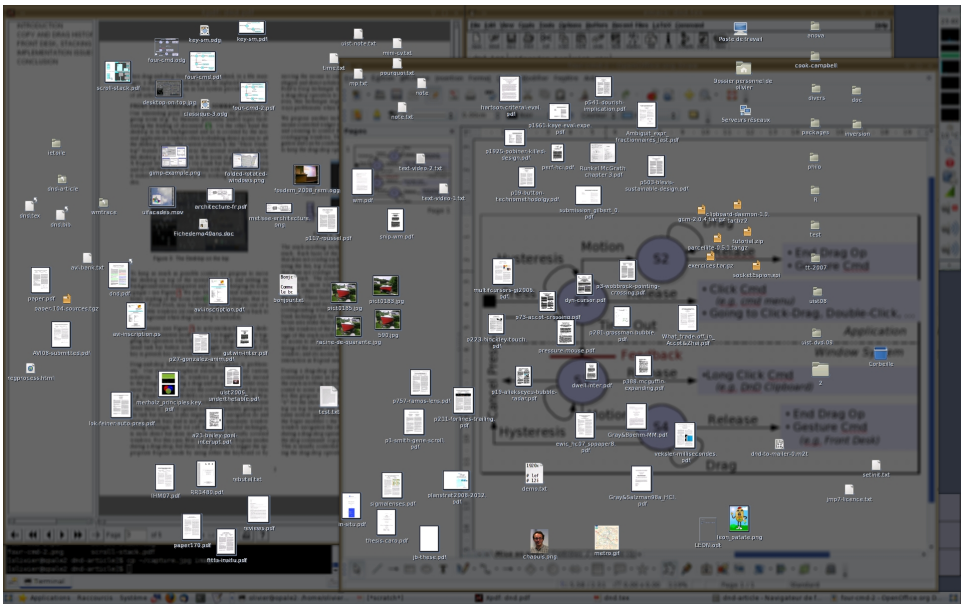


FIGURE 2.5 – DeskPop *: le bureau est placer au dessus de tout autre objet. En appliquant deux effet graphique : une semi-transparence et un flou, l'utilisateur dispose d'un aperçu du bureau et des fenêtres au même instant.*

est un espace depuis toujours considéré comme spécial.

Le *Bureau* est le plus souvent utilisé comme une aire de stockage temporaire permettant un accès rapide à un certain nombre de ressources. Les icônes peuvent y être regroupées afin de créer des motifs facilitant la recherche visuelle [94]. Cependant une limitation du *Bureau* est qu'il est affiché en permanence à l'arrière-plan. De ce fait, il est le plus souvent recouvert, au moins partiellement, par des fenêtres cachant alors des icônes. Pour faire face à ce problème, les environnements de bureau mettent généralement à disposition de l'utilisateur une commande "Show Desktop" qui cache ou déplace temporairement les fenêtres afin de rendre le *Bureau* visible. Ce type de solution n'est cependant pas idéal pour nous puisqu'il requiert un changement important de contexte, à savoir la disparition temporaire de toutes les fenêtres.

Notre technique *DeskPop* est conçue de façon à accéder à l'espace et au contenu du *bureau*, tout en préservant au maximum le contexte spatial. Cette interaction place le bureau au premier plan. Cependant, afin que l'utilisateur conserve une vue sur ses fenêtres maintenant recouvertes par le bureau, le fond du bureau devient semi-transparent, permettant de voir les fenêtres derrière celui-ci. De plus, le fond du bureau agit comme un verre dépoli par l'application d'un effet de flou. Cet effet permet au contenu du bureau (typiquement des icônes et les textes les accompagnant) de rester lisibles et facilement distinguable [11] des fenêtres en arrière-plan

et de leur contenu. La Figure 2.5 illustre un cas réel de l'utilisation de la technique *DeskPop*.

L'activation de *DeskPop* est réalisée à l'aide du mécanisme de délai présenté dans la section 2.2 et utilise un appui long sur le bouton droit de la souris suivi d'un geste (Figure 2.2, bas). Deux comportements de *DeskPop* sont accessibles par deux gestes différents selon l'intention de l'utilisateur. Dans la version par défaut lorsque l'utilisateur amorce une opération de glisser-déposer, *DeskPop* se désactive afin que l'utilisateur puisse de nouveau interagir avec les fenêtres. Cependant, l'utilisateur peut également demander à ce que la technique reste active jusqu'à ce qu'il la désactive lui même. Ce dernier comportement est nécessaire si l'utilisateur souhaite par exemple réorganiser son bureau. La version permamente est déclenchée, après un appui long sur le boutton droit, par le geste "L" alors que la transciente l'est par le geste "⌟".

Une telle technique peut améliorer les opérations de glisser-déposer depuis le bureau vers une fenêtre ouverte, permettant une meilleure utilisation de cet espace de stockage temporaire. Par exemple, un chercheur a lu un article intéressant quelques jours plus tôt et l'a sauvé sur son bureau pour se rappeler de l'envoyer à l'un de ses collègues qui pourrait également être intéressé. Il se décide finalement pour envoyer un courrier électronique. Il écrit son courrier et lorsqu'il veut inclure l'article, il effectue un clic droit long puis dessine "⌟" pour invoquer la version transiente de *DeskPop*. Il saisit alors le fichier correspondant sur le bureau et commence son mouvement de déplacement immédiatement vers la zone de dépôt des fichiers joints dans l'interface de son client de courrier électronique. Ayant utilisé la version transiente de l'interaction, *DeskPop* se désactive automatiquement et notre utilisateur peut déposer l'article dans son courrier et l'envoyer. La rapidité d'exécution de l'inclusion de l'article dans le courrier tient à deux choix de conception de *DeskPop*. Tout d'abord, la visualisation proposée par notre technique permet à l'utilisateur de commencer son déplacement directement dans la bonne direction et avec une bonne précision de mouvement puisqu'il voit l'emplacement de dépôt tout au long de l'interaction. Ensuite, l'aspect transient de la commande permet tout simplement de ne pas avoir à désactiver *DeskPop*.

2.5 StackLeafing

Comme exposé dans l'introduction de ce chapitre, il est difficile d'utiliser la technique conventionnelle de `glisser-déposer` dans les cas où la cible de la sous-action de déposer est occultée – la fenêtre ayant été iconifiée ou la zone de dépôt se trouvant sous une autre fenêtre.

Plusieurs solutions ont été imaginées dans les gestionnaires de fenêtre pour pallier à cette limitation.

Il existe par exemple des interactions basées sur l'utilisation de délais ou de pauses dans l'interaction. Lorsque l'utilisateur immobilise son curseur "tenant" un objet en cours de manipulation pendant un temps déterminé au-dessus d'une fenêtre, celle-ci passe au premier plan pour la rendre entièrement visible. L'utilisateur peut alors terminer son action de glisser-déposer. Cependant, ce type d'interaction n'offre aucune aide dans le cas d'une fenêtre "cible" totalement occultée.

L'interaction proposée par la fonction "Exposé" de Mac OS X résout ce problème grâce à deux modes différents permettant de rendre visibles simultanément soit les fenêtres de l'application courante soit toutes les fenêtres ouvertes. Cependant, retrouver la fenêtre désirée n'est pas toujours aisé. A mesure que le nombre de fenêtres augmente, Exposé les affiche sous forme d'images de plus en plus petites et de plus en plus éloignées de leur position d'origine. L'utilisateur se retrouve alors à devoir sélectionner une fenêtre parmi un ensemble de sosies difficiles à distinguer. De plus, encore une fois, l'utilisation d'Exposé requiert soit l'utilisation d'un raccourci clavier, obligeant l'utilisateur à accéder à celui-ci alors qu'il utilise la souris, soit un déplacement du curseur dans un des coins de l'écran, ce qui nécessite un déplacement conséquent.

Dans la plupart des systèmes, l'utilisateur peut aussi utiliser la barre de tâches (en glissant l'objet sur un élément de la barre pour mettre la fenêtre associée au premier plan). Cette barre des tâches se présente souvent comme une barre placée sur un des bords de l'écran où sont affichés une icône et/ou un texte décrivant chaque application ou fenêtre ouverte. Un point faible de cette technique est que l'espace dans une barre de tâche est limité. De ce fait il peut être difficile d'exposer une icône (avec un texte) pour chaque fenêtre ouverte. Ce problème a déjà donné lieu à des solutions techniques. La plus simple est de regrouper les fenêtres par application ou par ressemblance de nom. Du point de vue spatial, on peut arguer que ce type de techniques conserve la disposition puisque chaque icône reste au même endroit dans la barre. Cependant, plusieurs problèmes persistent. Tout d'abord, la plupart du temps ce n'est pas l'organisation spatiale qui est conservée mais la position d'une icône dans la liste représentée dans la barre. De ce fait, les icônes sont souvent redimensionnées selon leur nombre et les regroupements faits. Ensuite, ce type de techniques oblige l'utilisateur à déplacer son attention loin de son point d'intérêt.

L'utilisateur peut également utiliser une combinaison de touches (e.g., `Alt-Tab`) afin de faire "défiler" les fenêtres de manière séquentielle pour trouver celle qui l'intéresse. Dans le meilleur des cas, le système fait passer la fenêtre actuellement sélectionnée sur le dessus de la pile des fenêtres pour que l'utilisateur puisse l'examiner. Dans le pire des cas, le système propose à l'utilisateur une liste de nom ou d'icônes où il doit retrouver sa fenêtre. Entre ces deux extrêmes, une multitude de solutions ont été envisagées et implantées dans les gestionnaire de fenêtres. Elles ont toutes le problème d'un accès séquentiel au fenêtre, d'autant plus lent que la fenêtre est loin dans la liste.

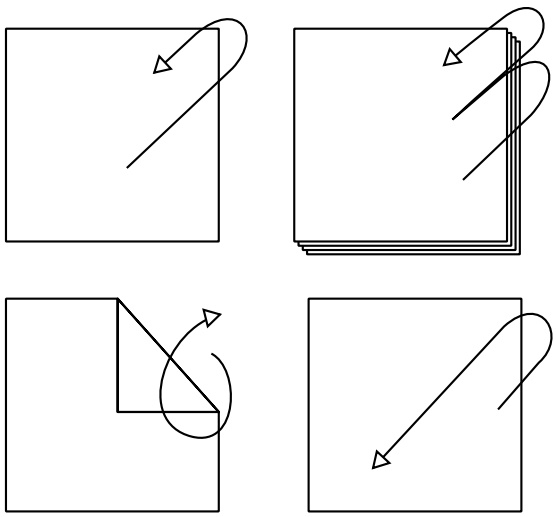


FIGURE 2.6 – *Les gestes utilisés par Fold n' Drop.*

Finalement, Fold'n Drop [32] propose une interaction originale permettant d'accéder aux zones occultés par "pliage" des fenêtres les recouvrant (voir la Figure 2.6). Les fenêtres sont pliées par des mouvements de franchissement sur leurs bords. La métaphore utilisée est celle de feuilles de papier qu'il est possible de plier à l'aide de l'index afin de feuilleter une pile de documents [12]. Malheureusement, ce type d'interaction requièrt des gestes pouvant être compliqués nécessitant parfois de la précision.

Comme nous le voyons, un certain nombre de techniques existent déjà pour permettre de sélectionner une fenêtre lors d'un glisser-déposer. Cependant la plupart de ces techniques souffrent d'un défaut qui oblige l'utilisateur à déporter son attention sur une autre partie de l'interface dans le but de faire son changement de fenêtre. Certaines de ces techniques requièrent également de parcourir les fenêtres une à une afin de trouver celle qui l'intéresse. Ce traitement entraine un nombre d'actions important lorsque le nombre de fenêtres est important.

Afin de concevoir une nouvelle technique d'interaction plus efficace, nous avons identifié trois pré-requis :

- Le déplacement de la souris doit être minimal. La taille des écrans est en augmentation, et il s'agit aussi de prendre en compte des configurations possédant plusieurs écrans. Demander à l'utilisateur d'effecteur de grands déplacements est une perte de temps et, de plus, le distrait de sa tâche ;
- Cette technique doit pouvoir être effectuée exclusivement à la souris ;
- Il faut conserver l'aspect de la manipulation directe autant que faire se peut.

Toutes ces considérations nous ont amenées à concevoir *StackLeafing*. Cette

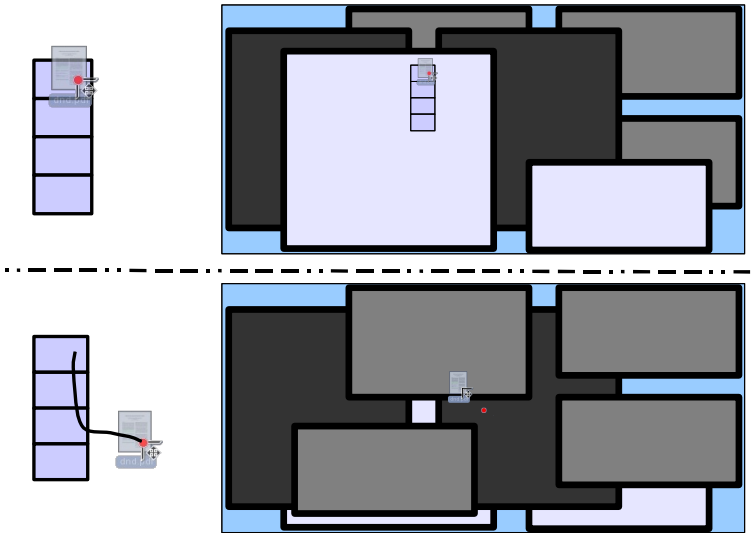


FIGURE 2.7 – *Un exemple de l'utilisation de la technique* StackLeafing. *L'utilisateur choisit de révéler la troisième couche de fenêtres en descendant de deux "cases" dans la pile puis en quittant la case par le côté.*

technique repose sur un objet graphique combinant le scrolling généralisé [108] et le franchissement [2] afin de naviguer entre **couches** de fenêtres (voir la Figure 2.7). Plus précisement :

- Lors du déclenchement de *StackLeafing* les fenêtres sont organisées en couches de fenêtres qui ne se recouvrent pas (Il y a 8 fenêtres mais 3 couches plus le bureau dans la Figure 2.7).
- Un widget, agrandi sur la partie droite de la Figure 2.7, permet la navigation entre les couches, chaque case correspondant à une couche : la navigation sur les cases permet de faire apparaitre la couche correspondante. Le bureau fait partie intégrante de ces couches et représente ici la dernière couche. Nous utilisons la technique *DeskPop* pour afficher le bureau.
- Pour continuer le glisser-déposer avec la couche afficher il suffit de quitter le widget par le côté (bas de la Figure 2.7).

StackLeafing est conçu pour être utilisé pendant une phase de glisser-déposer. Ce type d'interaction, malgré ses qualités, pose un problème pour déclencher d'autres interactions lorsque l'utilisateur se trouve dans le pseudo-mode qu'elles introduisent, c'est-à-dire pendant la phase de déplacement (glisser).

Une solution simple pour les concepteurs d'interfaces et d'utiliser un autre dispositif d'entrée comme le clavier pour déclencher les autres interactions. Cependant, si l'utilisation du clavier peut être une solution viable dans certains cas comme avec un ordinateur de bureau, une interaction plus directe à l'aide du dis-

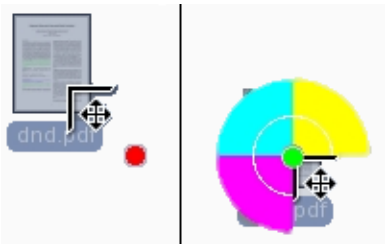


FIGURE 2.8 – Trailing Widget *: Le point rouge suit le curseur de la souris de manière élastique. Un mouvement brusque permet d'attraper le point, révélant un menu circulaire permettant de lancer des commandes.*

positif de pointage (souris, stylet, doigt) peut être plus utile dans d'autre cas (e.g., un dispositif mobile tel une tablette).

Nous avons aussi envisagé d'utiliser la molette de la souris pour naviguer dans les couches de fenêtres lors d'un glisser-déposer. On fait ainsi correspondre la molette avec le troisième degrés de liberté existant dans un gestionnaire de fenêtre (la profondeur) et on permet ainsi à la souris d'être en association $1:1$ avec l'espace qu'elle permet de contrôler [105]. Ceci dit, il ne nous a pas semblé que l'utilisation de la molette avec un bouton de la souris pressé soit idéal. De plus, tous les dispositifs de pointage ne disposent pas de molette.

Nous avons donc opté pour une interaction pouvant être réalisée entièrement à l'aide du dispositif de pointage grâce à l'utilisation d'un objet graphique poursuivant le curseur appelé "trailing widget" [41]. Notre implémentation prend la forme d'un point rouge suivant de manière élastique l'objet déplacé (Figure 2.8 partie gauche). L'utilisateur peut "attraper" cet objet graphique par un court et rapide mouvement dans sa direction. Attraper cet objet permet d'activer un menu circulaire dont trois quartiers permettent d'activer des commandes par simple franchissement (Figure 2.8 partie droite), dont celle de *StackLeafing*.

Le quartier du menu qui se trouve dans la direction utilisée pour attraper le "trailing widget" permet de désactiver le menu. La position de cette possibilité d'annulation de l'interaction est conçue pour permettre un comportement acceptable dans le cas d'une activation accidentelle. Les deux autres quartiers restants, permettent (i) d'activer la technique *DeskPop* pour un accès immédiat au bureau ; et (ii) de changer la sémantique du glisser-déposer entre déplacement et copie (généralement effectué par l'utilisation de la touche `Ctrl`).

```
windows ← listWindowsFrontToBack()
layers ← []

while (not empty?(windows) {
    currentLayer ← [unpop(windows)]

    foreach (currentWindow : windows) {
        if (not overlap?(window, currentLayer)) {
            currentLayer ← currentLayer ++ currentWindow
            windows ← windows -- currentWindow
        }
    }

    layers ← layers ++ currentLayer
}
```

FIGURE 2.9 – *L'algorithme de création des couches de fenêtres.*

2.6 Couches de Fenêtres

L'une des idées de *StackLeafing* est de permettre à l'utilisateur de voir chaque fenêtre distinctement sans changer sa connaissance spatiale de l'organisation des fenêtres afin de faciliter la recherche. Nous prenons également en considération qu'il est de plus en plus pénible de gérer les fenêtres une par une.

Nous avons donc fait le choix d'effectuer des regroupements de fenêtres. Ces regroupements se font de façon à ne pas déplacer les fenêtres dans le plan. Pour arriver à ce résultat, nous regroupons les fenêtres ne se recouvrant pas ensemble.

L'algorithme utilisé pour créer les couches de fenêtres fonctionne comme suit. Il considère les fenêtres selon leur ordre de superposition en allant de celle du dessus vers la plus profondes. Il crée une couche ne contenant tout d'abord que la première fenêtre. Il considère alors les fenêtres restantes en les ajoutant à la couche en cours de création si elles ne sont pas recouvertes par celles appartenant déjà à cette couche. Une fois toutes les fenêtres passées en revue et s'il reste encore des fenêtres non assignées à une couche, l'algorithme construit une nouvelle couche de façon similaire. Cet algorithme est détaillé à la Figure 2.9.

Une faiblesse de algorithme ci-dessus est qu'il n'est pas très stable aux modifications de la position d'une fenêtre dans l'ordre de profondeur. Toutefois, il est important de noter que cette technique a tout d'abord été conçue pour une interaction courte dans le temps : le glisser-déposer

Pour une utilisation plus large de ce type de regroupement, de nombreuses améliorations peuvent être proposées. Par exemple, Xu et Casiez dans Push-and-Pull Switching [120] ont ajouté une condition sur la quantité de recouvrement des

fenêtre permettant de regrouper des fenêtres qui auraient appartenues à des couches différentes avec notre algorithme.

2.7 Implémentation

L'implémentation des techniques de ce chapitre a été réalisée avec Métisse [26]. Métisse est un système de fenêtrage X Window expérimental qui est constitué d'un serveur X spécial qui dessine les fenêtres (une à une) dans des pixmaps et d'un gestionnaire de fenêtres (dont le code base provient de FVWM [6]) avec un module de rendu des fenêtres, *FvwmCompositor*, en OpenGL. *FvwmCompositor* se connecte au serveur X et peut récupérer le contenu des fenêtres, à la manière d'un visualiseur Xvnc, mais avec la différence notable que le système fonctionne fenêtre par fenêtre (alors que Xvnc fonctionne au niveau d'un écran virtuel). Les fenêtres sont rendues comme des polygones texturés et le compositeur avec le gestionnaire de fenêtres se charge de l'interaction avec l'utilisateur. Un point fort du système est sa grande souplesse en ce qui concerne la redirection, vers les applications, des évènement d'entrées [110].

Métisse (ici *FvwmCompositor*) permet donc d'intercepter et de transformer tous les évènements du pointeur et du clavier et de les renvoyer modifiés ou non aux applications (fenêtres). Ceci est nécessaire pour, par exemple, implanter la machine à états décrite dans la Figure 2.2. Le système de rendu de Métisse nous a aussi permis d'implanter la technique *DeskPop*. Avec Métisse le bureau n'est qu'une fenêtre comme une autre ce qui nous permet de la rendre au premier plan. De plus nous avons utilisé un shader GLSL pour modifier son rendu (comme dans la Figure 2.5). Pour laisser les icônes inchangées tout en modifiant le fond, nous avons imposé une couleur unie au fond, et ainsi nous pouvons obtenir la couleur du fond en choisissant un pixel qui n'est pas dans la boite englobante d'un icône (ces boites englobantes étant obtenues via l'interface d'accessibilité). En se qui concerne le flou sur les fenêtres, nous avons utilisé un autre shader que l'on applique aux fenêtres.

Outre l'utilisation de Métisse il nous a fallu utiliser d'autres services qui ne sont pas en général fournis par les boites à outils graphiques standards. Nous avons utilisé l'extension X Window *XFixes* pour détecter les opérations de sélections, de copies, de couper et de glisser-déposer. Il nous a aussi fallu observer le système de fichiers pour détecter les déplacements de fichiers afin de maintenir l'historique de glisser-déposer à jour. Le noyau de Linux propose un tel service via un système de notification appelé *inotify*.

6. http://www.fvwm.org

Toutes les interactions décrites dans ce chapitre sont utilisables dans une version non distribuée de Métisse. Elles ont été utilisées de façon quotidienne par un chercheur sur une période de plus de six mois, période pendant laquelle elles ont été perçues comme utiles, pratiques et agréables à utiliser.

2.8 Conclusion

Dans ce chapitre nous avons présenté un ensemble de techniques d'interactions dont le but est de conserver le contexte spatial de l'utilisateur et de garder l'interaction localisée autour du point d'intérêt de celui-ci. Ces techniques ont toutes été implémentées dans un prototype utilisable avec des applications réelles, ce qui nous a permis d'acquérir une bonne expérience de leur utilisabilité. Chacune de ces techniques mériterait une étude plus approfondie de son utilisabilité et de ses performances, par exemple en conduisant des expérimentations contrôlées et des études de terrain. Cependant, nous avons préféré approfondir des questions plus fondamentales qui sont liées à ces techniques. La suite de cette thèse traite trois de ces questions, résumées ci-dessous.

(1) Pourquoi garder les fenêtres visibles avec la techniques *DeskPop* ? Des comportements comme le fait de cacher les fenêtres pour les faire réapparaître ultérieurement ou le fait de pousser ces fenêtres vers le bord de l'écran puis de les remettre en place une fois l'interaction terminée ont-ils un impact sur les actions de l'utilisateur ? Cette question est étudiée dans le chapitre 3 où, de manière abstraite, nous comparons le pointage dans le cas des technique *Show Desktop, Exposé* et *DeskPop*.

(2) Les techniques *DeskPop* et *StackLeafing*, afin de conserver la disposition des fenêtres (dans le plan) se reposent sur la manipulation de la profondeur. Or cette profondeur dans les gestionnaires de fenêtres est une dimension encore assez peut étudié. Dans le chapitre 4 est présenté une étude d'indices visuels de profondeur ayant pour but de rendre cette dimension plus présente pour l'utilisateur.

(3) Afin de conserver l'utilisateur concentré sur la tâche qu'il est en train d'accomplir, les commandes de déclenchement des historiques de copier-ou-déplacer ont été conçues en utilisant non pas une interaction spatiale mais temporelle. Cette idée d'utiliser une notion duale de l'espace afin de ne pas perturber celui-ci est développée dans le chapitre 5 avec la développement d'un concept d'interaction rythmique.

CHAPITRE 3

Pointage sur Cibles Animées ou Surgissantes

3.1 Introduction

La souris informatique fut inventée par Douglas Englebart pour le système NLS, puis fut un élément important lors du développement des interfaces supportant la métaphore du bureau virtuel tel que le Xerox Star puis les systèmes Apple comme le Lisa ou le Macintosh. Depuis lors, l'acquisition de cibles à l'écran à l'aide d'un dispositif de pointage (ou tâche de pointage) est devenue l'une des activités fondamentales et l'une des plus fréquentes lors de l'utilisation d'interfaces graphiques. L'introduction de nouveaux matériels comme les dispositifs mobiles tactiles ne change pas cet état de fait : la base de l'interaction avec ces systèmes reste inchangée, simplement l'utilisateur pointe avec son ou ses doigts plutôt qu'avec une souris. Une preuve supplémentaire de l'importance du pointage est la littérature conséquente existant en interaction homme-machine sur ce sujet. La majorité de ces recherches entre dans trois catégories :

- La conception de nouvelles techniques afin d'améliorer ou de faciliter le pointage ;
- La comparaison de différents dispositifs de pointage et de leurs performances ;
- La conception de nouveaux modèles (ou l'adaptation de modèles existants) pour estimer le temps nécessaire au pointage et acquérir une meilleure compréhension de la tâche dans différentes situations.

Cependant, la plupart de ces recherches se focalisent sur le cas de *cibles statiques*. Nous qualifions de *statique* les cibles dont l'affichage est antérieur au début du pointage. Ce cas particulier, bien que courant, est le cas de base de toute la recherche sur le pointage en IHM, qui fait abstraction, la plus souvent, de la diversité des situations que l'utilisateur est amené à rencontrer lors de l'utilisation d'interfaces réelles. Dans ce chapitre nous étudions le pointage sur des cibles non statiques, à savoir des cibles animées ou surgissantes dont la position est connue

de l'utilisateur, comme par exemple lorsque l'on sélectionne une commande dans un menu déroulant ou un menu pop-up.

L'outil majeur dans l'étude du pointage est la loi de Fitts [39], qui permet de prédire le temps nécessaire (en moyenne) pour l'acquisition d'une cible d'une taille `W` placée à une distance `D` du point de départ. Cette loi provient du domaine de la psychologie expérimentale, mais a été adaptée pour la recherche en interaction homme-machine. La forme la plus couramment utilisée est celle présentée par Mackenzie [76] et dont l'expression est :

$$MT = a + b \log_2 \left(\frac{D}{W} + 1 \right)$$

Dans cette expression, MT est le temps moyen pour acquérir la cible et a et b sont des constantes déterminées empiriquement et dépendantes du dispositif. Le terme $\log_2 \left(\frac{D}{W} + 1 \right)$ est appelé *l'indice de difficulté* de la tâche de pointage (ID dans le reste du document). Malgré le nombre conséquent d'adaptations proposées de cette loi (voir [91] pour une revue), aucun consensus n'existe pour l'instant en psychologie quant à l'explication de cette loi.

L'interprétation la plus populaire de la loi de Fitts au sein de la communauté de recherche en IHM est celle avancée par Meyer et ses collègues avec le modèle des sous-mouvements [80] : Lors d'un pointage un premier sous-mouvement rapide mais peu précis est effectué en direction de la cible, cette phase de pointage est souvent appelé phase balistique. Par la suite, si cette première phase n'a pas permis d'atteindre la cible, plusieurs sous-mouvements successifs, souvent plus petits, sont effectues jusqu'à l'acquisition finale de la cible. L'une des caractéristiques intéressantes de ce modèle des sous-mouvements, pouvant expliquer l'introduction d'un logarithme dans l'expression de la loi de Fitts, est que la courbe donnant la vitesse de déplacement par rapport à la distance parcourue ressemble souvent à une cloche possédant un pic de vitesse marqué.

Les interfaces graphiques ont toujours proposé des cibles *surgissantes* ("pop-up" en anglais), c'est-à-dire dont l'apparition à l'écran est postérieure au début du mouvement de pointage. Un utilisateur expert, familiarisé avec le fonctionnement de son interface développe une mémoire motrice du placement des objets à l'écran. Il peut donc arriver qu'il commence à diriger son pointeur vers la position où il pense que sa prochaine cible va apparaître afin de gagner du temps. Un exemple courant d'un tel comportement peut être observé lors de l'usage d'un menu surgissant (menu pop-up) ou de boîtes de dialogues familières.

Les menus sont l'exemple emblématique des cibles surgissantes. Ils ont fait l'objet d'une attention particulière de la part des chercheurs en IHM. Dans le cas

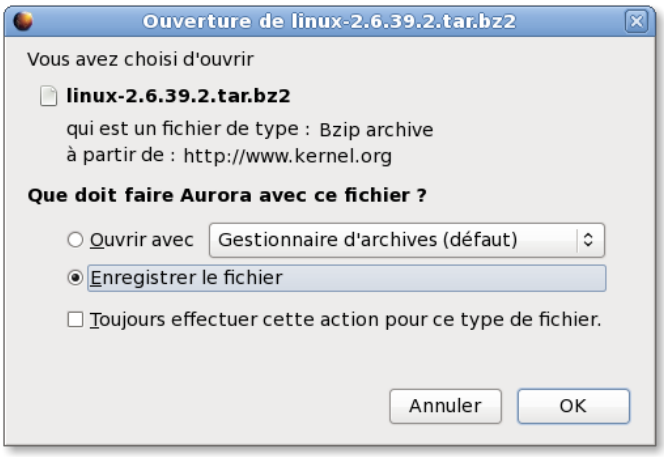


FIGURE 3.1 – *Boîte de dialogue du navigateur internet Firefox pour l'enregistrement d'un document.*

des menus linéaires, il est communément admis que les menus apparaissent instantanément. L'étude de ces menus s'est intéressée à des paramètres comme le nombre d'éléments composant le menu ou la profondeur d'imbrication des sous-menus. Pour sa part, [29] étudie la navigation dans les menus et sous-menus en utilisant conjointement la loi du mouvement canalisé [1] qui peut modéliser le passage d'un menu à un sous-menu, la loi de Fitts et la Loi de Hick-Hymann [52, 61, 103] qui prédit le temps nécessaire pour trouver un élément dans un ensemble (ordonné).

Dans le cas de menus plus évolués comme les *Marking Menus* [71] et leurs nombreuses variantes, on distingue deux modes :

- un mode novice dans lequel l'utilisateur attend un court laps sans déplacer le curseur afin de faire apparaître (surgir) le menu.
- un mode expert dans lequel le menu n'apparaît pas à l'écran.

Dans ce type de menu, le délai est donc utilisé pour faire la distinction entre les deux modes, novice et expert (voir aussi [55]).

Les boîtes de dialogue sont un autre exemple de cibles surgissantes ou animées pour lesquelles l'utilisateur peut avoir une connaissance préalable de la position. Ce type de fenêtres apparait souvent au même endroit et comporte peu de boutons (typiquement "Ok" et "Annuler") qui constituent les cibles potentielles. Les utilisateurs savent très souvent quelle action ils veulent faire bien avant l'apparition de cette boîte de dialogue (depuis l'interaction précédente avec le système). Nous avons observé à l'aide d'un outils d'enregistrement de l'interaction dans les gestionnaires de fenêtre [24] que les utilisateurs avaient en effet tendance à anticiper l'apparition des boîtes de dialogue en commençant le déplacement du curseur vers le bouton désiré avant que celui-ci n'apparaisse. Un exemple représentatif de ces boîtes de dialogues est la fenêtre demandant à l'utilisateur ce que le navigateur internet Firefox doit faire quand l'utilisateur clique sur un lien menant à un

document non HTML (voir la Figure 3.1). Du fait des délais dans les connexions réseau, ce type de boîte de dialogue peut apparaître avec des délais imprévisibles, cependant les utilisateurs arrivent à anticiper leur apparition. Enfin sous Mac OS X (et d'autre applications comme les nouvelles versions de Firefox), les boîtes de dialogues standard se déroulent sous la barre de titre des fenêtres. Elles apparaissent donc toujours centrées en haut de la fenêtre courante et la position des boutons est donc aisément prévisible par les utilisateurs.

Un autre exemple intéressant de cibles que l'on peut considéré comme surgissantes se retrouve lors de la navigation des pages internet. Les utilisateurs visitent souvent les mêmes pages [86] et connaissent souvent leur agencement. Souvent, ils suivent également le même chemin de navigation [114]. Par exemple, l'utilisateur charge une page, clique sur le bouton d'identification (les champs sont pré-remplis par le navigateur), puis il clique sur un lien vers la page désirée. Le temps de chargement et de rendu des pages dépend non seulement de la connexion au réseau, mais aussi de l'algorithme de rendu du navigateur et de la structure des pages. Comprendre les effets de ces délais sur les tâches de pointage pourrait avoir des conséquences sur la conception des navigateurs et des pages internet et plus généralement sur celui des interfaces graphiques.

Plus récemment, grâce à l'augmentation de la puissance de calcul des ordinateurs grand public, de plus en plus d'interfaces graphiques proposent des animations. Les animations ont de nombreuses applications dans les interfaces graphiques utilisateur, allant d'un usage à vocation esthétique à des usages permettant à un utilisateur de mieux interpréter des données ou résoudre des problèmes [45] ou de mieux comprendre les effets de ses actions sur le système d'information par l'ajout d'un retour visuel continu de l'état du système [101, 100]. Certaines de ces animations peuvent avoir lieu en parallèle d'une tâche de pointage. Nous savons de manière empirique que les utilisateurs sont capables de se servir de leurs connaissances du système pour déduire la trajectoire et possiblement la position finale des cibles animées. Cependant l'effet de cette animation sur le pointage n'a fait à notre connaissance l'objet d'aucune recherche systématique.

Ainsi, nous n'avons pas trouvé de littérature concernant le pointage sur une cible "invisible". L'inverse, cependant, à savoir le pointage à l'aide d'un curseur invisible, à été abondamment étudié (voir par exemple [19]) afin de savoir si les mouvement du bras étaient soumis au modèle de "position" ou au modèle d'"amplitude". Il est intéressant de noter que l'étude décrite dans la suite de ce chapitre ferait plutôt pencher pour le modèle "amplitude".

En ce qui concerne les cibles animées, de nombreuses études en psychologie, comme [56], et plus récemment en IHM [84, 50] ont étudié l'acquisition de cibles animées. Ces études ont cependant étudiée une *tâche de capture* : l'acquisition de la cible se fait pendant sont déplacement. Dans notre cas, la capture ne peut se faire que lorsque la cible est immobilisé à sa destination finale. Comme autres

exemples d'interactions pouvant illustrer ce type de pointage, nous pouvons citer des techniques d'aide au pointage dont le principe de base est d'amener les cibles potentielles (éventuellement à l'aide d'animations) plus près du curseur après que le mouvement de pointage ait commencé : Drag-and-pop [10] et Vacuum [17] en sont des exemples. Par contre, ces techniques ne rendent pas aisé l'anticipation de la position finale des cibles.

3.2 Motivations dans le cadre de la thèse

La motivation principale de l'étude des cibles surgissantes et animées que nous présentons ci-dessous est la technique "DeskPop" présentée au chapitre précédent : la stabilité des fenêtres offerte par cette technique lui donne-t-elle un avantage par rapport aux techniques usuelles qui cachent ou animent les fenêtres. Nous voulions donc comparer notre technique à celle communément employées : Exposé d'Apple et la technique "Show Desktop".

Pour illustrer notre motivation prenons le scénario suivant :

Un utilisateur veut faire glisser une icône depuis le bureau pour la déposer dans la fenêtre courante. Cette icône est actuellement cachée par les fenêtres ouvertes à cette instant. C'est une opération usuelle par exemple pour attacher un fichier à un courrier électronique ou pour ouvrir un document. L'utilisateur doit alors déclencher une commande permettant d'afficher le bureau, acquérir l'icône à l'aide de la souris, retrouver la fenêtre voulue et la rendre de nouveau active, puis y déposer l'icône à l'endroit désiré.

- Si l'utilisateur se sert du système d'exploitation Mac OS X, il peut effectuer cette opération en faisant appel à la technique d'interaction Exposé pour rendre le bureau visible et accéder à l'icône. Toutes les fenêtres sont alors repoussées à l'extérieur de l'écran en utilisant une animation. Puis, lorsque l'utilisateur commence à déplacer l'icône, les fenêtres sont animées pour se replacer à leur position originale. Ce dernier cas correspond à une *cible animée* (condition *Anim* dans l'expérience ci-dessous) : l'utilisateur doit déposer l'icône à un endroit qu'il connaît déjà, mais la fenêtre contenant cette position est animée.
- Dans le cas de Windows et de la plupart des environnements de bureau offerts par le système X Window, l'utilisateur à la possibilité d'utiliser la fonction "Show Desktop" qui masque ou icônifie les fenêtres ouvertes pour montrer le bureau. On peut alors saisir l'icône désirée puis faire réapparaître les fenêtres à leur emplacement précédent pour terminer l'interaction. Ce cas correspond à une *cible surgissante* (condition *Popup* de notre expérience) : l'utilisateur sait où déposer l'icône mais la fenêtre contenant cette position apparaît plus tard, soudainement.

– Enfin, l'interaction que nous avons conçu dans le cadre de Power Tools correspond à une *cible statique* (condition *Static* de notre expérience) : Le bureau est placé au-dessus des fenêtres, mais celles-ci restent visibles à travers le fond du bureau et sont donc visibles de l'utilisateur avant que celui-ci ne commence le déplacement de l'icône.

Cependant, par souci de généralisation, nous ne voulions pas nous limiter au cas décrit ci-dessus. Nous avons donc abstrait la tâche ci-dessus pour notre étude afin de couvrir plus largement les exemples de cibles surgissantes ou animés cités précédemment.

En comparaison des études précédentes sur le pointage, l'originalité de notre étude est donc que la cible peut être initialement invisible lorsque la tâche débute et peut bouger lors du pointage. La majorité des travaux de recherche existants sur le pointage ne prennent en compte que le cas particulier dans lequel la cible du pointage est visible tout au long de la tâche, y compris au début. Une exception notable est le travail de Cao et collègues [21] dans lequel la cible n'est pas visible au début de l'interaction mais est découverte par la manipulation d'une "fenêtre" de visualisation suivant le curseur (curseur couplé). Dans ce cas, le pointage peut-être découpé en deux phases : tout d'abord trouver la cible puis l'acquérir. Dans notre étude, la cible apparaît automatiquement (sans répondre à une action de l'utilisateur) et l'utilisateur connait par avance l'emplacement final de la cible. Comparé à l'étude de Cao, nous nous attendons donc dans notre cas à un mouvement plus fluide comportant des adaptations dynamiques selon le comportement de la cible.

En effet, des études ont montré que le temps le plus court pour que des informations du monde physique captées par le système sensoriel aient un effet sur le système moteur est d'environ 100 *ms* [63]. En particulier, Flash et Henis [40] ont observé des ajustements de la trajectoire d'un mouvement compris entre 100 et 200 *ms* après la modification par le système de la trajectoire du curseur lors d'une expérience utilisateur. Plus proche de la recherche en interaction homme-machine, Zhai et collègues [121] ont montré que les utilisateurs étaient capables de tirer avantage du grossissement *non prévisible* d'un cible lors d'un pointage, suggérant une dynamique d'adaptation rapide du mouvement.

La connaissance préalable du point d'arrivée ou d'apparition de la cible est un point important de notre étude. Ce type d'interaction à fait l'objet d'une étude dans le cas des menus linéaires [57]. Cependant, notre étude est plus générale, prenant en compte plus de cas d'utilisations, et se concentre plus spécialement sur le paramètre de temps d'apparition (`DELAY`). La mémorisation d'une cible unique dans la mémoire à court terme à été étudiée dans le contexte de la vision humaine ([4]) et montre que les êtres humains parviennent à se remémorer l'emplacement d'objets dans l'espace par saccades (rapides mouvements de l'oeil) à partir d'un petit ensemble de positions mémorisées.

3.3 Expérience

L'expérience présentée dans cette section sur les cibles animées ou surgissantes est, pour des raisons d'abstraction et de contrôle, réduite à des tâches de pointage à une dimension. La loi de Fitts est essentiellement un modèle à une dimension et l'étude en deux dimension aurait demandé l'introduction d'autres facteurs [3] qu'il était difficile d'inclure dans cette étude sans en faire exploser sa complexité.

3.3.1 Matériel

L'expérience a été réalisée sur un Macbook Pro possédant un processeur Core2-Duo cadencé à 2.33 Ghz et pourvu d'une carte graphique ATI Radeon X1600 dont la sortie était présentée sur un écran LCD de 24 pouces à une résolution de 1920x1200 pixels. Le dispositif de pointage était une souris optique Logitech RX 250 possédant une résolution de 1000 dpi. Les réglages par défaut d'accélération du pointeur de Mac OS X d'Apple étaient appliqués. L'expérience a été réalisée en Java à l'aide de la bibliothèque *SwingStates* [5].

Les animations étaient des transitions linéaires calculées à un taux de 60 images par seconde.

3.3.2 Participants

Douze volontaires adultes non rémunérés (dix hommes et deux femmes) ont participé à cette expérience. Tous étaient droitiers et âgés de 23 ans à 35 ans (pour une moyenne de 27,6 ans et une médiane à 26 ans). Tous les sujets étaient des utilisateurs expérimentés de l'outil informatique et étaient familiers avec le pointage à la souris.

3.3.3 Tâche et procédure

Les essais sont des tâches d'acquisition de cible en une dimension qui se déroulent de la façon suivante :

– Tout d'abord, une cible verte apparaît à l'écran (Figure 3.2.a). Le sujet doit cliquer dessus pour indiquer qu'il est prêt à effectuer l'essai

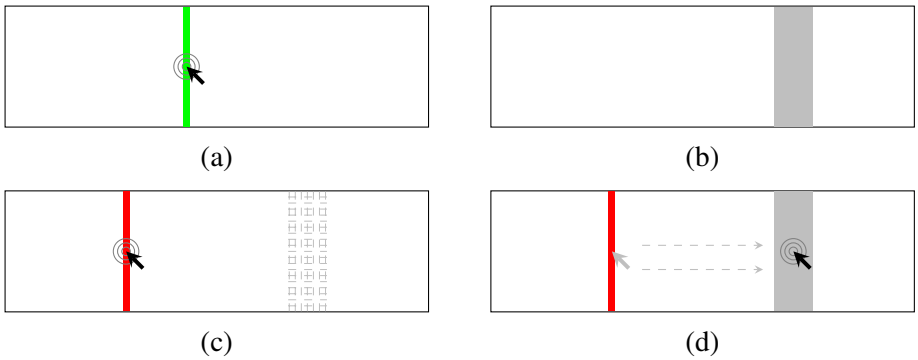


FIGURE 3.2 – *(a) Début de la procédure d'essai : cliquer sur la cible verte ; (b) Phase de mémorisation : affiche la cible principale pendant une seconde ; (c) La cible principale reste affichée (*Static*) ou est cachée (autres conditions) : cliquer sur la cible rouge pour débuter la phase de pointage ; (d) fin de l'essai : cliquer sur la cible principale (après que l'animation ait cessée ou que la cible soit apparue, selon la condition).*

– La cible verte disparaît alors et la cible (grise) principale de l'essai apparaît à l'écran à sa position finale (celle qu'elle aura à la fin de l'animation ou quand elle réapparaîtra) pendant une seconde (Figure 3.2.b) pour que l'utilisateur puisse mémoriser (acquérir la connaissance) de la position de la cible de son pointage : c'est la phase de mémorisation.
– Après la disparition de la cible principale, une cible rouge apparaît (Figure 3.2.c). La suite de l'essai dépend alors de la condition dans laquelle se trouve le sujet : Dans la condition *Static*, la cible principale reste affichée à l'écran alors qu'elle disparaît dans toutes les autres conditions. Indépendamment de la condition, le participant doit alors cliquer sur la cible rouge afin de faire débuter la phase de pointage proprement dite et marquer le début l'enregistrement du temps de mouvement. Selon la condition la cible principale va alors avoir un comportement différent :
  Static : La cible principale reste inerte jusqu'à la fin de la phase d'essai ;
  Anim : La cible principale est animée d'un mouvement fluide depuis le bord de l'écran le plus proche de sa position finale vers cette position. Cette animation dure `DELAY` *ms* ;
  Popup : La cible principale apparaît après `DELAY` *ms* à sa position finale.

La phase d'essai prend fin lorsque le sujet parvient effectivement à acquérir la cible principale lorsque celle-ci se trouve à sa position finale (voir la Figure 3.2.d) .

Le temps de mouvement des participants est enregistré depuis le moment où ils relâchent le bouton de la souris sur la cible rouge jusqu'au moment où ils pressent le bouton de la souris sur la cible principale. Si un sujet clique hors de la cible

où tente d'acquérir la cible avant que celle-ci soit arrivée à sa position finale (e.g., avant qu'elle ait fini son animation), l'essai est considéré comme raté et comptabilisé compté comme une erreur. Toutefois, en cas d'erreur de la part du participant, l'essai continue jusqu'à ce que la cible principale soit correctement acquise. Il était demandé aux participants d'effectuer la tâche le plus rapidement possible tout en faisant le moins d'erreurs possibles. Pour empêcher les participants de mémoriser la position finale de la cible principale à l'aide du curseur, celui-ci est caché lors de la phase de mémorisation.

3.3.4 Protocole

Cette expérience a été conçue en utilisant un protocole intra-sujet partiel $2 \times 5 \times 3 \times 3$ dont les facteurs sont les suivants :

- 2 conditions sur les techniques (`Tech` : *Anim* et *Popup*) ;
- 5 conditions de délai (`DELAY` : *Static* et 0, 200, 350, 500 millisecondes). Selon la condition expérimentale, ce facteur représente la durée pendant laquelle la cible est animée jusqu'à son emplacement final ou le temps après laquelle elle apparaît à cette même position ;
- 3 largeurs de cibles différentes (`W`) pour la cible principale : 16, 32, 64 pixels ;
- 3 distances d'acquisition (`D`) entre les centres de la cible rouge et celui de la cible principale lorsque celle-ci se trouve à son emplacement final : 256, 512 et 768 pixels.

Le protocole n'est que partiel (non complètement factoriel) car le croisement des facteurs `Tech` et `DELAY` ne donne que $2 + 2 \times 3 = 8$ conditions puisque les conditions `DELAY` *Static* et 0 sont équivalentes pour les deux conditions du facteur `Tech` : une animation de 0 ms fait apparaître la cible immédiatement à sa position finale. *Static* n'est pas vraiment une condition de `DELAY`, mais il est toutefois intéressant et pratique de la considérer à la fois comme condition de référence (puisque c'est la condition utilisée dans les études de Fitts classiques) et comme cas extrême du facteur `DELAY` (il est à noter que de ce fait, le facteur `DELAY` devient nominal, mais il sera plus tard de nouveau possible de le considérer comme continu).

Cette expérience utilise comme valeur médiane de `DELAY` 350 ms car c'est une valeur rencontrée fréquemment dans les interfaces graphiques comme par exemple pour Exposé. Une expérience préliminaire a été réalisée pour trouver deux autres valeurs formant une fourchette raisonnable pour des temps d'animation ou de Popup : 200 *ms* pour la borne inférieure et 500 *ms* pour la borne supérieure.

Une instance de l'expérience comporte deux parties. La première, de deux blocs : un avec le facteur `DELAY` $=$ *Static* et le second avec ce même facteur $= 0$. La seconde partie de l'instance correspond au croisement des facteurs `Tech` (*Anim*

et *Popup*) et des conditions restantes de DELAY (200, 350, 500 *ms*) : 2×3 blocs. Les blocs sont créés selon le facteur Tech et un carré latin de 3 par 3 est utilisé pour contre-balancer ce facteur en le croisant avec les trois conditions positives de DELAY entre les participants. Ceci donne 6 possibilités que nous croisons avec les deux premiers blocs. Pour ce faire, nous avons séparé les 12 participants en deux groupes de 6. Chacun de ces groupes utilise les 6 combinaisons possibles mais le premier groupe commence par la condition DELAY $=$ *Static* alors que le second débute par la condition DELAY $= 0$.

Chacun des blocs décrit ci-dessus est composé de 7 réplications des 3×3 combinaisons possibles des facteurs D et W, présentées dans un ordre aléatoire. Le première bloc est considéré comme un bloc d'apprentissage et n'est pas pris en compte dans les enregistrements. Les participants pouvaient prendre une pause au début de chaque réplication.

En résumé, le nombre total d'essais enregistrés au cours de l'expérience est de 8 blocs $\times$ 9 combinaisons de largeurs par distances $\times$ 6 réplications $\times$ 12 participants $=$ 5184 essais. Nous avons donc enregistré 72 essais pour chacune des conditions complètes, 6 pour chaque sujet. L'expérience a duré entre 42 et 56 minutes (pour une moyenne de 48 minutes et une médiane à 47 minutes).

3.4 Résultats

Dans cette analyse, le temps de mouvement MT mesure le temps jusqu'à ce que le participant presse le bouton de la souris avec succès sur la cible principale (et non pas à la première pression sur le bouton de la souris). Ceci permet de prendre en compte partiellement les erreurs d'acquisition dans le temps de mouvement.

Les données correspondant aux conditions *Static* et 0 du facteur DELAY ont été dupliquées afin de simuler ces conditions à la fois pour *Popup* et pour *Anim*. Ceci permet d'utiliser la méthode d'analyse de variance sur mesures répétées entièrement factorielle classique :

$$\texttt{MT} \sim \texttt{Tech} \times \texttt{DELAY} \times \texttt{D} \times \texttt{W} \times \text{Random}(\texttt{PARTICIPANT}).$$

Les temps de mouvement de plus de deux fois l'écart type du temps de mouvement moyen (par PARTICIPANT, Tech, DELAY, D et W) sont considérés comme étant des mesures aberrantes. Ces mesures, représentant 0,35% des enregistrements, ont été écartées de cette analyse. Toutefois, inclure ces mesures aberrantes donne des résultats très similaires à ceux présentés dans le reste de cette section.

Factors	DF	DFDen	F	p	
Tech	1	22	4,66	$0,0539$	(n.s)
DELAY	4	44	97,74	$< 0,0001$	(*)
D	2	22	68,63	$< 0,0001$	(*)
W	2	22	481,94	$< 0,0001$	(*)
Tech × DELAY	4	44	1,43	$0,2396$	(n.s.)
Tech × D	2	22	0,19	$0,8303$	(n.s.)
Tech × W	2	22	1,05	$0,3653$	(n.s.)
DELAY × D	8	88	8,33	$< 0,0001$	(*)
DELAY × W	8	88	1,90	$0,0686$	(*)
D × W	4	44	2,67	$0,0444$	(*)
Tech × DELAY × D	8	88	0,19	$0,9920$	(n.s.)
Tech × DELAY × W	8	88	0,35	$0,9436$	(n.s.)
Tech × D × W	4	44	0,51	$0,7274$	(n.s.)
DELAY × D × W	16	176	1,11	$0,3469$	(n.s.)
Tech × DELAY × D × W	16	176	1,05	$0,4071$	(n.s.)

TABLE 3.1 – *Résultats de l'ANOVA pour* $MT \sim Tech \times DELAY \times D \times W \times$ *Random(*$PARTICIPANT$*).*

Le tableau 3.1 montre les résultats d'une analyse de variance répétée sur le temps de mouvement. Comme attendu, les facteurs habituels du pointage que sont la distance à la cible et sa largeur (D et W) ont tous deux un effet significatif important sur le temps de mouvement : plus la tâche est difficile, plus il faut de temps pour la réaliser (voir la Figure 3.3). Le facteur Tech pour sa part n'a pas d'effet significatif sur le temps de mouvement et il ne présente également aucune interaction avec les autres facteurs. La différence moyenne de temps entre les conditions *Anim* et *Popup* n'est que de 17 *ms* avec un avantage pour *Popup*. Cette différence ne représente que 2% du temps de mouvement moyen. Encore une fois, la duplication des enregistrements pour les conditions *Static* et $DELAY = 0$ n'ont pas d'influence sur ces résultats statistiques puisqu'ils s'annulent mutuellement.

Concernant le facteur DELAY, il a un effet significatif important sur le temps de mouvement (voir la Figure 3.4). Une analyse à post-hoc utilisant un test de Tukey ($\alpha = 0.05$) ne montre, en moyenne, aucune différence significative ni entre *Static* et $DELAY = 0$ ni entre $DELAY = 0$ et $DELAY = 200$. Par contre cette même analyse révèle une différence significative entre les conditions *Static* et $DELAY = 200$ (74 *ms* en faveur de la condition *Static*, une amélioration de 7.7%). Elle montre également que la condition $DELAY = 200$ est significativement plus rapide

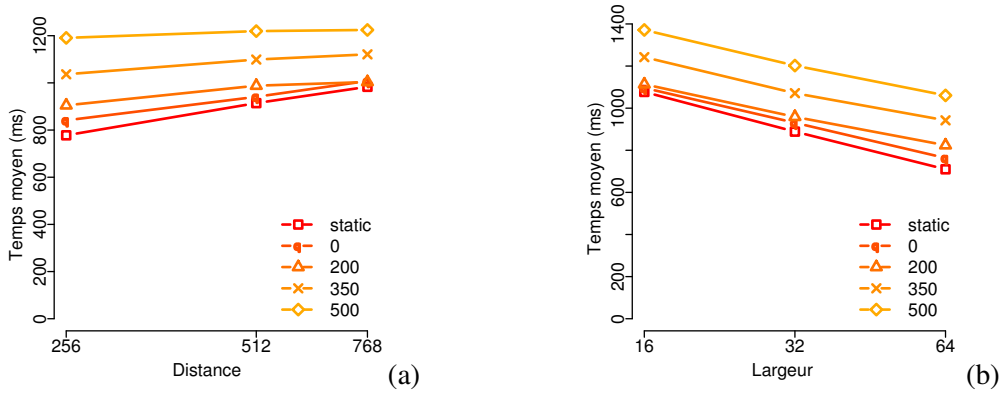


FIGURE 3.3 – *(a) Temps de mouvement en fonction de la distance pour chaque conditions de* `DELAY`*. (b) Temps de mouvement en fonction de la largeur de la cible principale pour chaque conditions de* `DELAY`*.*

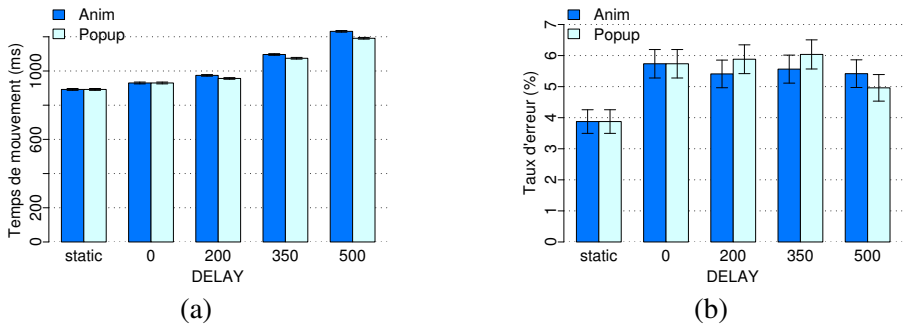


FIGURE 3.4 – *(a) Temps de mouvement pour chacune des conditions de* `DELAY` *et* `Tech`*. (b) Taux d'erreur pour chacune des conditions de* `DELAY` *et* `Tech`*.*

que `DELAY` $=$ 350 (120 *ms*, amélioration de 11.1%) et que `DELAY` $=$ 350 est significativement plus rapide que `DELAY` $=$ 500 (126 *ms*, amélioration de 10.4%).

Il existe également une interaction significative entre les facteurs `DELAY` et `D` (voir la Figure 3.3.a). Cette interaction suggère que la distance à moins d'effet lorsque le `DELAY` s'accroît (de *Static* et `DELAY` $=$ 0 à `DELAY` $=$ 500). De fait, une analyse post-hoc utilisant un test de Tukey ($\alpha = 0.05$) montre qu'il existe une différence significative entre chaque distance pour *Static* et `DELAY` $=$ 0, une différence significative entre les distances 256 et 512 mais pas entre 512 et 768 pour `DELAY` $=$ 200, une différence significative entre les distances 256 et 768 mais pas entre 256 et 512 ni entre 512 et 768 pour `DELAY` $=$ 350 et, finalement, aucune différence significative pour `DELAY` $=$ 500.

La Figure 3.3.a suggère également que les différences en temps de mouvement qui existent entre la conditions *Static*, `DELAY` $=$ 0 et `DELAY` $=$ 200 diminuent

lorsque la distance augmente. En effet, une analyse a posteriori utilisant un test de Tukey ($\alpha = 0.05$) révèle une différence de temps de mouvement significative entre les conditions *Static* et DELAY $= 200$ pour une distance de 256 pixels, mais aucune différence pour une distance de 768 pixels.

Une explication probable de ce phénomène est que les participants commencent leur mouvement de pointage dès qu'ils ont relâché le bouton de la souris lors de l'acquisition de la cible rouge, c'est-à-dire avant la fin de l'animation de la cible principale ou son apparition à sa position finale. Leur premier sous-mouvement, balistique, est alors plus précis lorsque l'animation cesse ou que la cible apparaît pendant que les participants sont toujours en train d'effectuer ce premier sous-mouvement, leur laissant la possibilité de le corriger si besoin est. Par exemple, pour DELAY $= 500$ et D $= 256$, les participants sont capables d'amener le curseur de la souris aux environs de la position finale de la cible – ou tout du moins la position qu'ils envisagent comme étant celle où la cible va apparaître ou se stabiliser – avant que celle-ci ne s'y trouve, ils doivent alors attendre son apparition ou son arrivée pour être en mesure de faire les derniers ajustements requis pour finir le mouvement de pointage. Au contraire, avec une condition telle que DELAY $= 200$ et D $= 768$, la cible se trouve à son emplacement final bien avant la fin du sous-mouvement balistique, laissant la possibilité aux participants d'ajuster la trajectoire de ce sous-mouvement pendant son exécution, le rendant ainsi plus précis. De plus, le fait d'attendre l'apparition de la cible ou sa stabilisation à sa position finale entraîne probablement une charge cognitive plus importante puisque l'utilisateur doit reconnaître ce fait puis initier de nouveau son mouvement.

Il existe également une faible interaction significative entre les facteurs DELAY et W. La Figure 3.3.b suggère que la différence entre *Static* , DELAY $= 0$ et DELAY $= 200$ croît lorsque la largeur de la cible augmente. Encore une fois, une analyse post-hoc utilisant un test de Tukey ($\alpha = 0.05$) confirme qu'il existe une différence de temps de mouvement significative pour ces trois conditions pour W $= 64$ alors qu'elle n'est pas significative pour la condition W $= 16$.

En ce qui concerne les erreurs d'acquisition, elles représentent 5.25% des essais enregistrés, un taux typique pour une expériences de pointage. La Figure 3.4.b montre ce taux d'erreur en fonction de la valeur prise par le facteur DELAY pour chaque valeur du facteur Tech. Une analyse répété de variance sur le taux d'erreur :

$$\texttt{ErrorRate} \sim \texttt{Tech} \times \texttt{DELAY} \times \texttt{D} \times \texttt{W} \times \text{Random}(\texttt{PARTICIPANT})$$

ne permet de déceler aucun effet significatif simple. En particulier, il n'existe pas de différence significative entre les conditions *Static* et DELAY $= 0$ bien que les taux d'erreurs pour ces conditions soit respectivement de 3.87% et de 5.73%. Par contre,

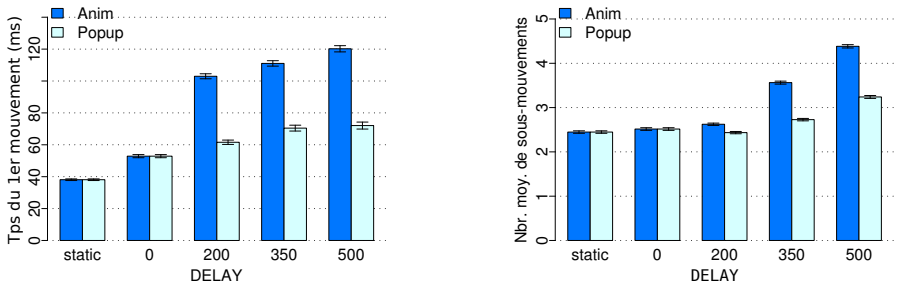


FIGURE 3.5 – *(a) Temps avant le premier mouvement en fonction de* `DEALY` *et* `Tech`*. (b) Moyenne du nombre de sous-mouvements en fonction de* `DELAY` *et* `Tech`*.*

un test logistique de Pearson 2 à 2 [1] fait apparaître une différence ($\chi^2 = 4.880$, $p = 0.0272$ pour $\texttt{DELAY} = 0$) entre *Static* (taux d'erreur de 3.87%) et les délais restants (taux d'erreurs entre 5% et 6%). Il n'y a pas de différence significative de taux d'erreur entre les autres combinaisons de `DELAY`. D'autres tests de Pearson réalisés pour les autres facteurs n'ont révélé aucun autre effet significatif sur les taux d'erreur.

En résumé, mis à part l'influence de la distance et de la taille de la cible sur le temps de pointage moyen, classique dans les expériences de pointage, cette analyse nous a permis d'identifier plusieurs facteurs importants dans le pointage sur des cibles surgissantes ou animées. Tout d'abord, et chose surprenante, au niveau statistique le comportement de la cible n'a pas d'effet sur le temps de pointage moyen. Concernant le délai d'animation ou d'apparition de la cible, s'il est inférieur à 200 *ms*, ce délai n'a pas d'influence significative sur le temps de pointage. Toutefois, lorsque ce temps augmente (supérieur à 200 *ms*), les performances de pointage se dégradent. Plus intéressante encore est l'interaction entre la distance de la cible et son délai d'animation ou d'apparition. Cette interaction suggère que si la cible n'est pas encore à sa position finale un peu avant la fin du mouvement balistique initial, la précision de ce mouvement est moindre, obligeant les participants à effectuer des mouvements de correction plus importants. Finalement, le cas statique engendre significativement moins d'erreurs de pointage que les cibles animées ou surgissantes (quel que soit le délai).

3.5 Analyse Cinématique

Une analyse plus poussée du mouvement de pointage montre que les participants commencent, comme attendu, leurs mouvements avant que la cible se trouve

1. Un test adapté à des mesures "discrètes".

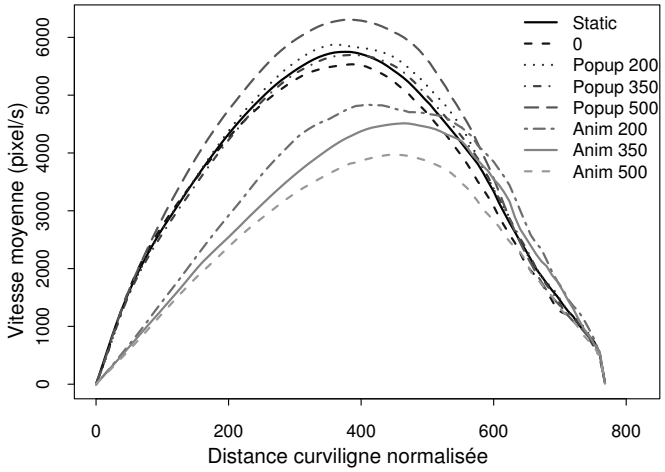


FIGURE 3.6 – *Vitesse moyenne en fonction de la distance curviligne normalisée pour* `D` *= 768 et pour chaque condition* `DELAY` $\times$ `Tech`. *(les courbes grises possédant un pic de vélocité moindre sont celles correspondant à la condition* Anim*).*

à sa destination finale. La Figure 3.5.a montre le temps pris par les participants après le relâchement du bouton de la souris sur la cible rouge et leur premier mouvement de souris. Ces temps sont clairement plus courts que `DELAY` lorsque ce dernier est positif. De plus, une analyse de variance ne montre aucune différence significative entre les différentes valeurs de `DELAY` pour la condition *Popup* et pour des délais différents de zéro dans la condition *Anim*. Cependant, comme le montre la figure 3.5.a ce temps avant le premier mouvement est significativement plus long pour les conditions `Tech` = *Anim* et `DELAY` > 0 que pour toutes les autres conditions ($\sim 40ms$).

Une analyse des enregistrements cinématiques du mouvement montre que les conditions *Popup* et *Anim* engendrent des profils cinématiques différents (voir la Figure 3.6 pour un exemple tiré d'essais avec `D` $= 768$, des profils similairent apparaissent pour les autres conditions). Ces profils ne semblent pas varier lorsque l'on change `W`, les enregistrement ont donc été regroupés en moyenne selon ce paramètre. Les courbes de vitesse en fonction de la distance pour les conditions *Popup* sont similaires à celle que l'on obtient dans les conditions *Static* et `DELAY` $= 0$: elles possèdent une forme de cloche usuelle avec un pic de vélocité aux environs de 50% de la distance à la cible. Les courbes correspondant à la condition *Anim* sont différentes, avec leurs pics de vélocité légèrement déportés vers la droite à 60% de la distance parcourue (ces différences peuvent être démontrées comme significatives statistiquement). De plus, dans cette condition *Anim* les mouvements sont plus lent avec un pic de vélocité moindre au fur et à mesure que `DELAY` augmente, phénomène non apparent dans le cas de la condition *Popup*.

Une explication possible de ces phénomènes est que, dans la condition *Anim*,

les participants essaient de suivre la cible ou sont distraits par son mouvement, entraînant la diminution observée de la vitesse de déplacement. Au contraire, dans la condition *Popup*, les participants déplacent leur curseur directement à l'emplacement mémorisé de la cible principale (sur la Figure 3.6.a, ce mouvement est même plus rapide pour `DELAY` $= 500$, ce qui n'est pas le cas pour les autres délais).

Afin de confirmer l'affirmation ci-dessus et pour mieux comprendre les phénomènes prenant place à la fin du mouvement de pointage, nous avons calculé le nombre de sous-mouvements pour chaque essais ne donnant pas lieu à une erreur. Ce calcul se fait en comptant le nombre de fois où le pointeur ne bouge pas pendant au moins 50 *ms* (pour prendre en compte une vitesse nulle) entre le pic de vitesse et la pression finale sur le bouton. Ce compte est augmenté de 1 pour prendre en considération le mouvement final. La Figure 3.5.b montre la valeur moyenne du nombre de sous-mouvements en fonction de `Tech` et `DELAY`. Une analyse de variance démontre un effet significatif de `Tech` et `DELAY` ainsi qu'une interaction significative entre ces deux facteurs. La condition *Anim* engendre plus de sous-mouvements que *Popup*, la différence d'environ un sous-mouvement étant significative pour un `DELAY` supérieur à 200 millisecondes. Pour leur part, `D` et `W` ont également un effet significatif sur le nombre de sous-mouvements. Pointer sur des cibles de faible largeur requiert plus de sous-mouvements et il faut également plus de sous-mouvements pour des cibles se trouvant à une distance `D` = 512 pixels et 768 pixels que pour des cibles se trouvant à 256 pixels. Toutefois, ces facteurs n'ont pas d'interaction avec `Tech` et `DELAY`.

Pour résumer les résultats exposés dans cette section, bien que les statistiques sur les performances entre le pointage sur des cibles surgissantes (*Popup*) et le pointage sur des cibles animées (*Anim*) ne montre pas de différences significatives, les mouvements effectués par les participants pour réaliser ces tâches de pointage le sont. Les mouvements pour une tâche de pointage sur une cible surgissante sont plus proches de ceux typiquement observé dans le cas d'une cible statique que pour le cas d'une cible animée. Alors que le mouvement de pointage dans la condition *Popup* semble suivre un modèle d'amplitude simple ("sauter" à la position anticipée de la cible puis corriger le mouvement lorsque celle-ci est finalement apparue), le mouvement dans la condition d'animation semble bien plus complexe et diffère du cas traditionnel observé lors du pointage sur une cible statique.

3.6 Étendre la loi de Fitts

Dans cette section nous voulons, tout d'abord, utiliser la loi de Fitts pour mieux comprendre les effets du facteur `DELAY` sur le temps de mouvement. La Table 3.2 présente les paramètres de la loi de Fitts et les r^2 ajustés pour l'ensemble des

DELAY	a	b	adj. r^2	# pts.
ALL	454	135	0,4998	45
Static	180	174	0,9656	9
0	299	153	0,9074	9
200	462	120	0,8973	9
350	599	116	0,7974	9
500	731	115	0,6654	9

TABLE 3.2 – *Paramètres de la loi de Fitts* $MT = a + b.ID$

données et pour chaque DELAY. Les temps de mouvement des tests étaient regroupés en moyenne sans prendre en compte ni les répétitions, ni les participants ni les différentes valeurs du facteur Tech puisqu'il a été montré précédemment que ces facteurs n'ont pas d'effet significatif sur le temps de mouvement (MT). Autrement dit, l'ensemble des données concernant chacune des combinaisons des facteurs DELAY, D et W sont combinés dans une moyenne. Comme on pouvait s'y attendre étant donné les résultats de l'expérience exposés dans les deux sections précédentes, la régression de l'ensemble des données à une loi de Fitts donne des résultats médiocres du fait de l'effet important du facteur DELAY (voir figure 3.7 (a)). Par contre, la régression à une loi de Fitts pour chacune des valeur de DELAY donne de très bons résultats pour *Static* (cas normal d'un pointage de Fitts) mais se dégrade au fur et à mesure que la valeur de DELAY augmente. Cette régression pour chacune des conditions de DELAY peut être améliorée en utilisant la formulation de Welford de la loi de Fitts : $MT = a + b.log_2(D) + b.log_2(W)$ (r^2 au alentours de 0,95), mais ce modèle ne parvient pas à modéliser avec assez de précision l'ensemble des données ($r^2 = 0,5448$).

Dans l'optique de trouver un modèle de régression incluant le facteur DELAY, il est nécessaire de créer un facteur continu TV comme suit : *Static* est associé à la valeur 0 *ms*, puis DELAY $= 0$ est associé à 100 *ms*. Finalement, les autres valeurs de DELAY sont associées à leur propre valeur. Ce palier de 100 *ms* est introduit car ce temps correspond au délai minimum observé avant que des informations sensorielles aient un effet sur le système moteur [63].

Muni de cette valeur continue TV, il est maintenant possible d'étudier le modèle suivant : $MT = a + b.ID + c.TV$. Celui-ci procure des résultats qui améliorent fortement la régression linéaire sur toutes les données :

$$MT = 307 + 135.ID + 0,64.TV$$

avec un r^2 ajusté de 0,8921. Ce résultat peut encore être amélioré en prenant le modèle de Welford comme inspiration :

$$MT = a + b.log_2(D) + c.log_2(W) + d.TV$$

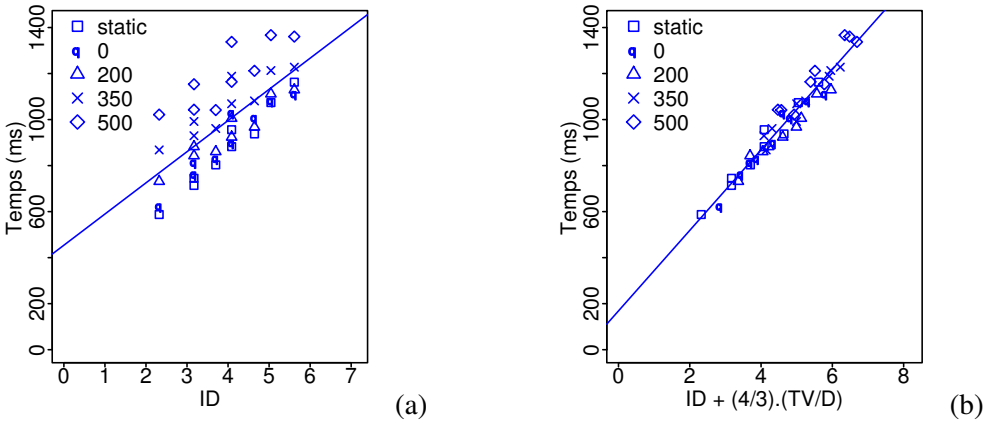


FIGURE 3.7 – *(a) Régression en prenant comme modèle la loi de Fitts (b) Régression en utilisant un ID modifié comprenant un terme $\frac{TV}{D}$.*

ce qui donne la régression suivante :

$$MT = 977 + 74.log_2(D) - 158.log_2(W) + 0,64.TV$$

avec un r^2 ajusté de 0,9478. Finalement, l'ajout d'un terme $log_2(D) * TV$ permettant de prendre en considération l'interaction entre les facteurs `D` et `DELAY` et améliore légèrement, mais significativement, la régression avec un r^2 ajusté de 0,9665.

Cependant, malgré l'amélioration des régressions que procurent ces modèles aucun n'est particulièrement intuitif. Il prennent simplement en considération le fait que `DELAY` a un effet sur le temps de mouvement et qu'il existe une interaction entre la distance à la cible et ce même délais. De plus, avoir plus de trois variables $(a, b, c, \dots)$ pose un problème de consistance avec la loi de Fitts puisqu'un seul facteur principal a été rajouté : `DELAY`.

Il a été noté précédemment que le temps de mouvement s'accroît avec `TV`, mais cette augmentation dépend de la distance séparant la cible et le point de départ du mouvement : lorsque la distance augmente, l'accroissement du temps de mouvement pour une même valeur de `TV` et moindre. De ce fait, l'effet combiné des facteur `TV` et `D` sur le temps de mouvement peut être exprimé par le ratio $\frac{TV}{D}$. Ce qui amène naturellement à étudier le modèle suivant :

$$MT = a + b.ID + c.\frac{TV}{D}$$

La régression des données avec ce modèle donne :

$$MT = 171 + 174.ID + 237.\frac{TV}{D}$$

accompagné d'une bonne valeur du r^2 ajusté : 0.9414.

En tenant compte que $\frac{237}{174} \sim \frac{4}{3}$, on peut intégrer le délai d'apparition dans avec l'indice de difficulté de Fitts pour définir un nouvel indice de difficulté :

$$ID' = ID + \frac{4}{3}.\frac{TV}{D}$$

dont on peut voir le tracé sur la Figure 3.7.b.

Il est évident que la validité de ce modèle doit être étudiée plus avant. Il peut avoir besoin d'ajustement pour prendre en compte un ensemble de distances plus grandes ou des cibles possédant des tailles différentes ou d'autres délais. Par exemple, un point crucial de la conception de cette expérience est que les valeurs utilisées pour le facteur `DELAY` sont à priori inférieures au temps de mouvement attendu dans les conditions classiques (la condition `Static`). De toute évidence, si `TV` est supérieur au temps de mouvement dans les conditions normales, un autre modèle doit être utilisé puisque l'utilisateur devrait alors déplacer le curseur vers la position mémorisée puis attendre l'apparition de la cible (après `TV` *ms*) et finalement acquérir la cible par un pointage classique. Ceci pourrait conduire à un modèle du type

$$MT = TV + a + b.\log_2(\frac{err(D)}{W} + 1)$$

où err est une fonction modélisant l'erreur de distance lorsque l'utilisateur pointe vers l'emplacement mémorisé d'une cible invisible à une distance `D`.

3.7 Conclusion

Nous avons présenté le premier travail sur l'acquisition de cibles animées ou surgissantes. Aucune différence significative n'a été trouvé entre les performances d'acquisition d'une cible statique et celle de l'acquisition d'une cible apparaissant immédiatement à une position mémorisé. Cependant, le cas statique engendre moins d'erreurs d'acquisition. Cette expérience montre également que le temps de pointage sur une cible statique est plus court que pour des cibles animées ou surgissantes avec un délais de plus de 200 *ms*. Ceci démontre que même de petits retards ou des animations dans une interface utilisateur peuvent détériorer les performances de pointage. Des techniques comme celles conçues lors de *Power Tools* (chapitre précédent) qui conservent le placement et l'affichage des cibles potentielles sont donc à préférer lorsque ceci est possible.

Nous n'avons pas trouvé de différence de performance significative entre l'acquisition de cibles apparaissant immédiatement et de cibles animées ou surgissant avec une délais de 200 *ms*. De grandes différences apparaissent toutefois lorsque

ce délai augmente (350 et 500 *ms* dans cette expérience), les écarts de temps observés étant consistant avec l'augmentation des délais. Ceci semble indiquer que les temps d'animations devraient être ramenés autour de 200 *ms* lorsque l'interaction requiert l'acquisition d'une cible à l'intérieur de l'objet animé. De plus les concepteurs d'interfaces graphiques devraient également essayer de restreindre la latence de l'apparition d'une cible potentielle en-deçà de 200 *ms*. Bien évidemment, dans les systèmes réels, d'autre facteurs, plus importants que la seule performance de pointage, sont souvent à prendre en considération comme par exemple la possibilité de détecter la causalité grâce aux animations.

Une autre découverte intéressante de cette étude est qu'elle suggère que les utilisateurs sont capables d'adapter leur mouvement de pointage dynamiquement en cours de mouvement. Cette adaptation semble dépendre de la nature de la cible : une animation affecte la nature même du mouvement, le rendant plus lent en vitesse de pointe et engendrant plus de sous-mouvements qu'une cible surgissante. Puisque l'expérience ne montre aucune différence significative entre les performances de pointage sur des cibles animées ou surgissantes (avec les même délais), il se peut que les utilisateurs soient capables de se servir de l'animation de la cible pour prédire plus finement la position finale de la cible. Ceci est à vérifier dans de futurs travaux.

Maintenant que les facteurs importants entrant en jeu ont été mieux compris grâce à une expérience abstraite, il s'agit de répliquer et d'étendre l'expérience avec d'autres facteurs qui rendent compte du contexte. Par exemple, il serait intéressant d'étudier le passage à une tâche de pointage en deux dimensions. On peut imaginer que dans ce cas des facteurs tel que la direction de l'animation et la largeur de la cible (vs cette direction) puissent jouer un rôle important. Il faudrait aussi répliquer l'expérience avec des tâches de pointage qui comportent de vraies fenêtres et icônes. D'autre part, les animations sont rarement purement linéaires, typiquement des ralentissements sont insérés au début et à la fin de l'animation pour les rendre plus agréables. Il serait donc intéressant d'étudier les effets de différents types d'animations pour des tâches de pointage sur des cibles animées. Finalement cette expérience repose sur l'hypothèse que les utilisateurs ont une bonne mémoire de l'emplacement final de la cible. Cette mémorisation peut elle aussi faire l'objet d'études.

CHAPITRE 4

Perception de la Profondeur en Gestion de fenêtres

4.1 Introduction

Dans le premier chapitre de cette thèse, nous avons présenté l'interaction *Stack Leafing* (voir section 2.5 et [37]). Cette technique de sélection de fenêtre lors d'un glisser-déposer a pour avantage de grouper les fenêtres en couches pour les parcourir plus rapidement. Une couche est formée par un regroupement de fenêtres selon un critère de non recouvrement. L'utilisateur peut ensuite naviguer de couche en couche pour accéder rapidement à la fenêtre cible. Cette interaction est intéressante car elle ne modifie pas la position des fenêtres dans le plan, permettant ainsi à l'utilisateur de tirer parti de sa mémoire spatiale pour l'aider à localiser la fenêtre désirée. La taille de la fenêtre est également laissée inchangée pour laisser son contenu lisible en toute circonstance. *Stack Leafing* ne repose que sur une modification de la profondeur des fenêtres.

Récemment, Xu et Casiez [120] ont également proposé d'utiliser le non-recouvrement des fenêtres pour faciliter leur sélection. La conception de leur technique, *Push-and-Pull Switching*, se base sur l'observation informelle que les utilisateurs placent les fenêtres participant à une même tâche de manière à ce qu'il y ait peu ou pas de recouvrement. Dans ces travaux, les couches de fenêtres sont en quelque sorte considérées comme des bureaux virtuels. Cependant, contrairement à des techniques de bureaux virtuels classiques, *Push and Pull Switching* ne force pas l'utilisateur à une relation 1: 1 entre fenêtre et tâche contrairement à *Window Scape* [116], ne force pas l'utilisateur à créer les tâches. Comme *Stack Leafing*, *Push-and-Pull Switching* utilise à leur avantage un comportement naturel des utilisateurs, qui répartissent dans l'écran les fenêtres qu'ils désirent accéder en même temps (définissant souvent ainsi une tâche [60]).

En résumé, là où la plupart des techniques de changement et de sélection de fenêtre pour les gestionnaires de fenêtres par chevauchement essaient en fait de contourner le problème de chevauchement, *Stack Leafing* et *Push and Pull Switching* essaient au contraire d'utiliser ce chevauchement et la façon dont les utilisateurs l'utilisent inconsciemment.

Cependant, ces nouvelles utilisations de la profondeur en interaction homme-machine appellent à une meilleure compréhension de cette dimension et en particulier, dans cette thèse, dans le cadre de la gestion de fenêtres. La perception de la profondeur dans une scène en deux dimensions, modèle proche de la gestion de fenêtres, est un processus complexe. Il existe en psychologie expérimentale une littérature conséquente sur ce sujet. En particulier, le modèle `FACADE` (Form-And-Color-And-Depth), développé par Grossberg et al. (voir [46] pour une introduction) a été créé pour comprendre les mécanismes permettant l'émergence de la perception de la profondeur (ou distance) dans les images ou les scènes en deux dimensions.

Il ressort de ces recherches plusieurs mécanismes permettant de faire apparaître une impression de profondeur. Parmi ceux-ci, les indices de profondeurs relatifs principaux sont ceux de nature géométrique comme le recouvrement partiel ou l'interposition d'un autre élément. Ce ne sont cependant pas les seuls. Dresp et al. [33, 49] montrent que les Indices Visuels de Profondeur (*IVP* dans la suite de ce chapitre) doivent être combinés de façon concordante afin que la profondeur soit perçue dans une scène à deux dimensions ("depth cue combination"). Les auteurs envisagent plusieurs IVP comme la luminosité ou la teinte d'un objet pour compléter les informations géométriques.

Toutefois, la perception de la profondeur dans le cadre de la gestion de fenêtres a fait l'objet de peu de recherches en interaction homme-machine. Le reste de ce chapitre présente l'une des premières études sur ce sujet. L'expérience contrôlée décrite ici explore l'influence de trois IVP (voir la Figure 4.1) sur la perception de la profondeur des couches de fenêtres.

Le premier IVP considéré est la luminosité : la luminosité générale d'une fenêtre diminue avec la profondeur de la couche à laquelle elle appartient. Cet indice reproduit un phénomène naturel de la vision : dans une scène possédant une seule source de lumière, plus les objets sont lointains, moins ceux-ci reçoivent de lumière et plus ils sont sombres.

Le second indicateur utilisé dans cette expérience est un effet de flou : plus la fenêtre d'intérêt se trouve en profondeur plus l'effet de flou qui lui est appliqué est puissant. Cet indicateur est également dérivé d'un phénomène optique simple : dans une scène, plus un objet est placé loin de l'observateur, moins ses détails sont perceptibles.

Le troisième est l'ombre : l'ajout d'une ombre aux fenêtres donne une indication relative de la position en profondeur d'une fenêtre par rapport aux fenêtres juste au-dessus ou en-dessous d'elle. Ainsi, cette condition ne représente pas exactement un IVP, mais renforce l'IVP géométrique.

Le but de l'expérience présentée dans ce chapitre est, premièrement, d'évaluer des indices permettant d'améliorer la perception de la profondeur dans le cadre

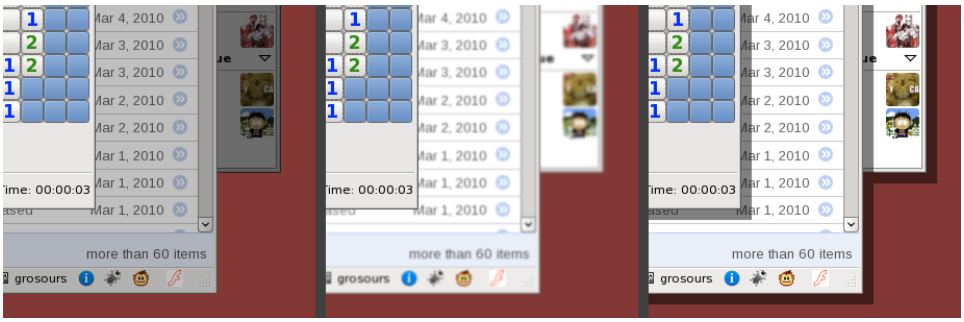


FIGURE 4.1 – *Les trois indices de profondeur testés : luminosité (à gauche), flou (au centre), ombre (à droite).*

de la gestion de fenêtres. La seconde ambition de cette étude est de vérifier si les utilisateurs sont capables de percevoir dans une scène complexe la totalité des couches de fenêtres dans le cadre d'une interaction similaire à *Stack Leafing* ou Push *and Pull Switching*.

4.2 La tâche

La tâche consiste à déterminer le nombre de couches présentes dans une scène. Chaque essai se déroule comme suit :

1. le sujet informe le système qu'il est prêt à commencer en pressant la touche espace du clavier ;
2. Une scène composée de `W` fenêtres réparties en `L` couches est présentée pendant 5 secondes au sujet ;
3. La scène disparaît et le système demande au sujet d'indiquer le nombre de couches qu'il a pu identifier dans la scène.

La répartition du nombre de fenêtres par couche est aléatoire. Cependant, afin de générer des répartitions ayant du sens, au moins $\frac{W}{2L}$ fenêtres sont affectées à chaque couche. Les couches sont créées à l'aide d'un algorithme permettant de placer des rectangles dans un plan sans que ceux-ci ne se recouvrent [15]. Lorsqu'une couche est terminée, nous créons des "ancres" (en pratique des carrés de 10 pixels de côté) que nous plaçons aléatoirement à l'intérieur des limites de chaque fenêtre de la couche. Au début de la création d'une nouvelle couche, les ancres de toutes les couches précédentes sont ajoutées à l'espace avant de commencer à placer les fenêtres de la couche, créant ainsi des zones où il est impossible de recouvrir les fenêtres se trouvant à des niveaux plus bas. Ces ancres garantissent donc qu'à la fin de la construction de la scène, toutes les fenêtres la composant sont visibles.

Ensuite, les indices de profondeurs éventuels sont appliqués à chaque fenêtre en fonction de la profondeur de la couche à laquelle elle appartient.

Le contenu des fenêtres est issu de captures d'écrans d'applications réelles représentatives des usages courants d'un ordinateur : navigateur Web, logiciel de messagerie, etc. De plus, l'algorithme peut placer certaines fenêtres partiellement en dehors des limites de l'affichage. En effet, il nous avons jugé important de reproduire une situation réaliste où un grand nombre de fenêtres possédant un contenu riche sont présentes, résultant en une forte charge perceptuelle pour les utilisateurs.

La tâche choisie, par contre, n'est pas une tâche réelle : un utilisateur se demande rarement quel est le nombre de fenêtres ou de couches affichées. Cependant, s'agissant d'une expérimentation sur la perception visuelle, une mesure quantitative aussi simple que le nombre d'éléments perçus semble la plus simple et la moins susceptible d'introduire un biais.

4.3 Sujets et Matériel Utilisé

Douze sujets (onze hommes, une femme) ont participé à cette expérimentation. Tous avaient une vue normale ou corrigée et sont des utilisateurs fréquents de l'ordinateur.

L'expérimentation était programmée avec le langage C++ et la bibliothèque graphique Clutter, une bibliothèque OpenGL optimisée pour la conception d'interfaces utilisateur en deux dimensions. Elle était exécutée sur un PC équipé d'un processeur Intel Core2 Duo 6700 cadencé à 2.66GHz, sous système d'exploitation Gentoo Linux. L'écran, de résolution 1920x1200 pixels, était commandé par une carte nVidia Quadro FX 3500 utilisant le pilote officiel de nVidia. Le clavier utilisé était standard.

4.4 Protocole Expérimental

L'expérimentation suit un plan intra-sujets complet $4 \times 4 \times 4$ dont les facteurs sont :

- 4 Indices Visuels de Profondeur (IVP) : *aucun*, *ombre*, *luminosité* et *flou* ;
- 4 nombres de couches (`L`) : 2, 3, 4, et 5 ;
- 4 nombres de fenêtres (`W`) : 5, 9, 11 et 15.

Les essais sont regroupés par blocs selon la condition IVP. La combinaison des deux autres facteurs (W et L, soit $4 \times 4 = 16$ cas) est ensuite présentée selon un ordre aléatoire. Chaque condition est répétée 4 fois par bloc, soit 64 essais par bloc. Enfin, la série de blocs est répétée une seconde fois afin d'atténuer l'effet d'apprentissage éventuel.

Chaque condition est donc répétée 8 fois soit $4 \times 4 \times 4 \times 8 = 512$ essais par participant. L'ordre de présentation des indices visuels de profondeur est contre balancé entre les sujets par un carré latin. Une session de 10 essais est présentée aux participants avant chaque bloc pour leur permettre de s'habituer à la condition d'indice visuel de profondeur.

Les deux mesures enregistrées sont :

- RT, le temps de réaction : c'est le temps mis par le sujet pour répondre à partir de l'affichage de la question ;
- ECART, la différence absolue entre la réponse attendue et la réponse de l'utilisateur : ECART $= 0$ signifie qu'il n'a pas fait d'erreur, alors que ECART $= 2$ indique qu'il a sous- ou sur-estimé de 2 le nombre de couches affichées.

4.5 Prédictions

Les prédictions suivantes résultent de notre interprétation de la littérature existante :

- P1 : *luminosité* et *flou* entraînent des écarts plus faibles entre les estimations des participants et le nombre réel de couches et un temps de réaction plus rapide que *ombre* et *aucun* car ce sont des indices absolus et continus.
- P2 : *ombre* entraîne des écarts plus faibles entre les estimations des participants et le nombre de couches réel et un temps de réaction plus rapide que *aucun* car un indice, même relatif, apporte de l'information.
- P3 : W a un effet significatif sur le nombre d'erreurs et le temps de réaction, car un plus grand nombre de fenêtres complexifie la tâche.

4.6 Résultats

Sur 6144 essais , 42 dont le temps de réaction était aberrant ont été éliminés, soit 0.68% des données. Sur les 6102 essais restant, 2248 comportent une erreur de prédiction du nombre de couches présentes dans la scène, soit un taux d'erreur de 36.7%. Le temps de réaction moyen est de 720 *ms*.

Les Tables 4.2 et 4.1 donnent les résultats de l'ANOVA sur mesures répétées

Source	*DF*	*DFDen*	*FRatio*	*Prob > F*
IVP	3	33,26	8,12	0,0003 (*)
L	3	32,17	7,87	0,0004 (*)
W	3	32,81	11,52	< 0,0001 (*)
IVP × L	9	99,11	1,65	0,1113 (n.s.)
IVP × W	9	97,61	1,42	0,1916 (n.s.)
L × W	9	98,07	3,42	0,0011 (*)

TABLE 4.1 – *ANOVA pour* RT ∼ *IVP* × L × W × *Rand*(*PARTICIPANT*)

Source	*DF*	*DFDen*	*FRatio*	*Prob > F*
IVP	3	33,24	11,72	< 0,0001 (*)
L	3	32,80	3,40	0,0289 (*)
W	3	33,13	51,86	< 0,0001 (*)
IVP × L	9	99,43	6,19	< 0,0001 (*)
IVP × W	9	97,92	1,87	0,0650 (n.s.)
L × W	9	97,94	23,91	< 0,0001 (*)

TABLE 4.2 – *ANOVA pour* ECART ∼ *IVP* × L × W × *Rand*(*PARTICIPANT*).

$$\text{IVP} \times \texttt{L} \times \texttt{W} \times \textit{Rand}(\textit{ PARTICIPANT })$$

pour les deux mesures ECART et RT (les interactions triples sont non significatives). Les comparaisons par paires présentées dans la suite utilisent le test post hoc HSD de Tukey ($\alpha = 0.05$).

IVP a un effet sur ECART et RT (Figure 4.2 et Figure 4.3). L'analyse post hoc montre que *luminosité* engendre des écarts significativement moins importants que tous les autres IVP et des temps de réaction significativement plus rapides que les autres IVP sauf *flou* (en moyenne des écarts 31% moins importants, 20% plus rapide que *ombre et aucun*, et 10% plus rapide que *flou*). Par contre, *flou* et *ombre* ne présentent pas de différence significative avec *aucun* pour ECART, et *flou* ne présente pas de différence significative avec *luminosité*, ni avec *aucun* et *ombre* pour RT.

La prédiction P1 est partiellement vérifiée : *luminosité* entraîne des écarts moins importants et est plus rapide que *ombre* et *aucun*, mais ce n'est pas le cas pour *flou*. A noter que, pour RT, *flou* se situe tout de même entre d'une part *luminosité* et d'autre part *ombre* et *aucun*.

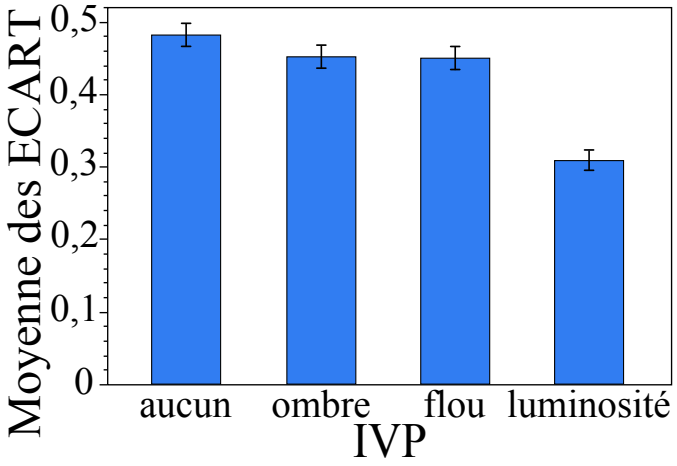


FIGURE 4.2 – *Moyennes des* `ECART` *en fonction des IVP.*

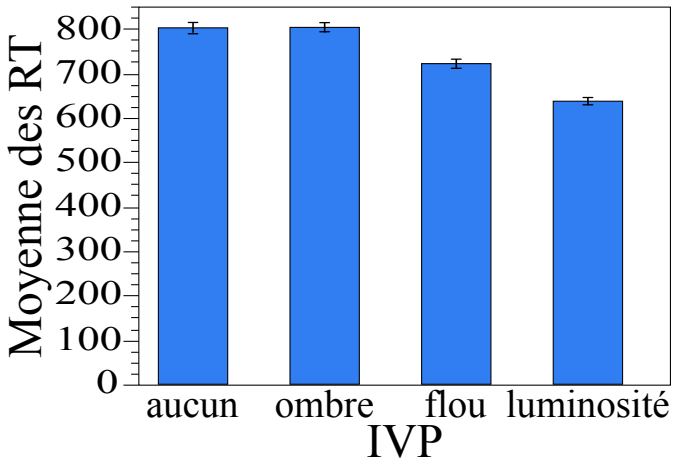


FIGURE 4.3 – *Moyennes des* `RT` *(ms) en fonction des IVP.*

P2 n'est pas vérifiée : les différences entre *ombre* et *aucun* ne sont pas significatives, que ce soit en temps ou en nombre d'erreurs.

`W` a un effet sur `ECART` et `RT` (Figure 4.4 pour `ECART`). Une analyse post hoc révèle que les sujets ont fait des écarts d'estimations significativement moins importantes et ont été plus rapides pour `W` = 5 que pour toutes les autres conditions (en moyenne des écarts 35% moins importants et 18% plus rapide). De plus, `W` = 15 engendre des écarts significativement plus importants que `W` = 11, 9 ou 5. Par contre, aucune différence significative n'a été trouvée entre les écarts moyens d'estimation correspondant à 9 ou 11 fenêtres et aucune différence significative n'a été trouvée entre les `RT` pour `W` = 9, 11 et 15. La prédiction P3 est donc partiellement vérifiée.

L a un effet sur ECART et RT (Figure 4.4 et 4.5 pour ECART). Une analyse post hoc sur ECART révèle que seuls $L = 5$ et $L = 3$ donnent des écarts d'estimation significativement différentes ($L = 3$ donnant des écarts 30% moins importants que $L = 5$). Pour leurs parts, $L = \{2, 4\}$ ne sont significativement différents d'aucun autre niveau de L. Cependant, une analyse post hoc sur RT indique que $L = \{2, 5\}$ sont significativement plus rapides que $L = \{3, 4\}$. Une explication possible pour ce résultat est que $L = \{2, 5\}$ sont les niveaux extrêmes et que les participants auraient sous-estimé leurs difficultés et les auraient donc traités plus rapidement, au prix d'erreurs d'estimation plus importantes pour $L = 5$.

L'interaction entre L et W est significative pour ECART et RT (Figure 4.4 pour ECART). En ce qui concerne RT l'interaction semble faible et n'influence pas les tests de différence en moyenne. Par contre, les sujets ont fait des écarts d'estimation significativement plus importants dans les conditions $L = \{2, 3\} \times W = 15$, que dans les autres L. Ceci pourrait indiquer que les participants ont été influencés par le nombre de fenêtres présentes : il leur a été difficile de percevoir qu'une scène comportant beaucoup de fenêtres avait peu de couches.

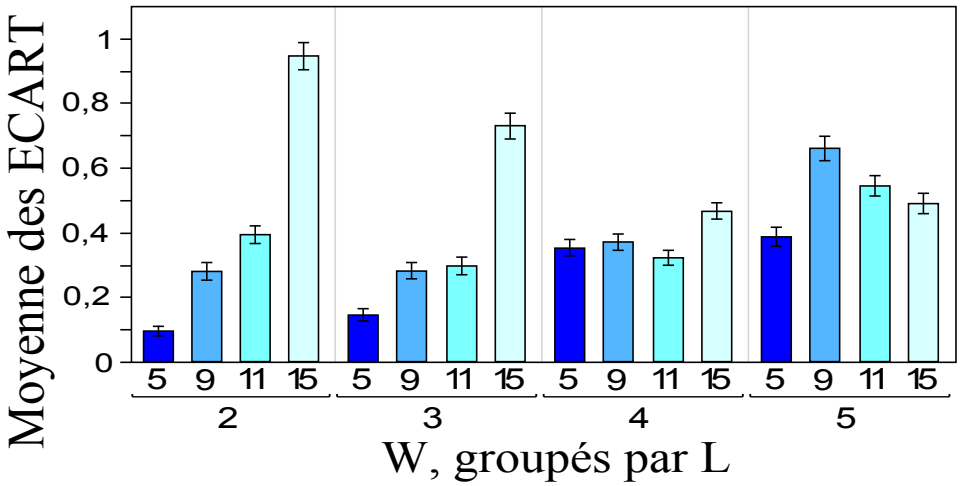


FIGURE 4.4 – *Moyenne des ECART en fonction de L et W.*

L'interaction entre L et IVP est significative pour ECART (Figure 4.5) mais pas pour RT. Une analyse post hoc montre que les écarts pour $L = \{2, 3, 4\}$ ne sont pas significativement différents pour toutes les conditions d'IVP. Cependant, pour $L = 5$ les différences entre *luminosité* et tous les autres IVP sont significatives (dans cette condition, *luminosité* engendre des écarts 54,5% moins importants que *flou*). Ce résultat peut s'expliquer par le fait que dans la condition *luminosité*, comme expliqué par un des participants, l'utilisateur peut se fier au niveau de l'IVP et "deviner" la profondeur de la couche la plus profonde.

En résumé, l'indice visuel de profondeur *luminosité* est le plus performant des trois testés car il engendre des écarts plus faibles dans les estimations du nombre de

couches par les participants. De plus, les performances de cet indice ne semblent pas se détériorer aussi rapidement que pour les autres indices lorsque le nombre de couches augmente. Enfin, les sujets sont capables de donner l'estimation plus rapidement avec l'indice *luminosité* qu'avec les autres indices visuels de profondeur. D'autre part, *flou* a des performances en retrait par rapport à ce que l'on pouvait attendre de cet indice. Enfin, il apparaît qu'il est difficile pour les participants de concevoir une scène ayant peu de couches, mais beaucoup de fenêtres.

4.7 Conclusion et Travaux Futurs

Dans ce chapitre, plusieurs indices visuels de profondeur ont été évalués dans le contexte de la gestion de fenêtres en couches. Le résultat principal de cette étude est que l'utilisation de la luminosité est efficace alors que le flou et l'ombre n'ont pas d'effet significatif sur la perception du nombre de couches. D'autre part les erreurs augmentent avec le nombre de fenêtres et avec le nombre de couches, mais c'est paradoxalement avec beaucoup de fenêtres et peu de couches que les erreurs sont les plus élevées.

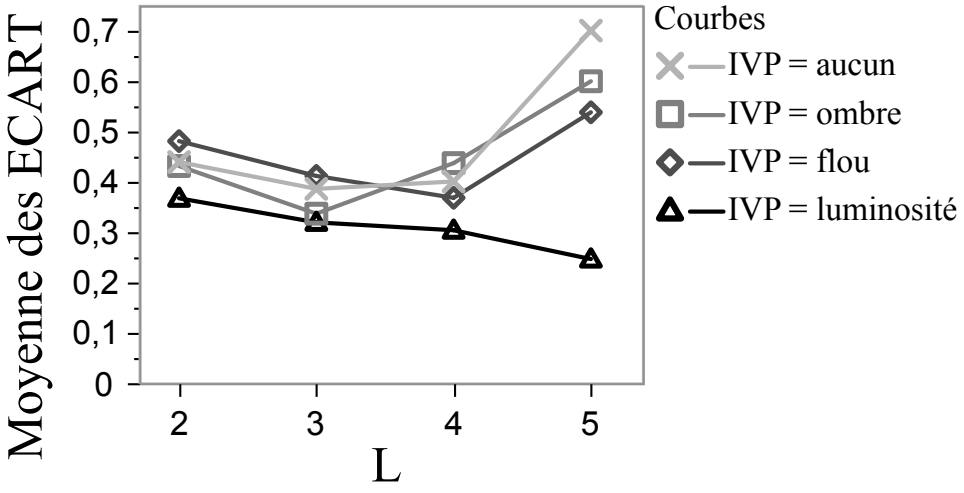


FIGURE 4.5 – *Moyenne des* `ECART` *en fonction de* `L`.

Le premier résultat donne une piste nouvelle et simple pour améliorer la perception de la profondeur des fenêtres : la luminosité. Il montre aussi que la technique la plus largement utilisée dans les systèmes actuels, l'ombre, ne semble pas aussi efficace que souhaitée. Par ailleurs, le fait que le flou ne semble pas non plus efficace est intéressant car c'est une technique coûteuse en ressources graphiques qui, de plus, rend le contenu des fenêtres moins lisible.

De nombreuses pistes restent à explorer au sujet des indices de profondeur. Par exemple d'autres IVP sont envisageables comme la translucence et la désaturation

des couleurs. La combinaison des différents IVP peut également être une piste intéressante.

L'un des problèmes possible des IVP est qu'ils peuvent diminuer la lisibilité du contenu des fenêtres sur lesquelles ils sont appliqués. Toutefois, il est possible de considérer la profondeur comme un indicateur du degré d'engagement [48] de l'utilisateur avec une fenêtre. Les utilisateurs interagissent généralement avec les fenêtres de premier plan au cours de la réalisation d'une tâche. Les fenêtres se trouvant au fond de l'écran, à moins de changement de tâche, ont donc peu de chance de représenter un intérêt pour l'utilisateur. Toutefois, il convient d'évaluer l'impact des IVP utilisés sur la lisibilité du contenu des fenêtres car s'il est peu probable que l'utilisateur ne désire visualiser le contenu des fenêtres les plus profondes, il semble évident qu'au moins les deux ou trois premiers niveaux de profondeur doivent rester facilement lisibles.

Enfin, ce chapitre n'étudie que des scènes statiques. Étudier les IVP dans un environnement dynamique où l'utilisateur change à la fois la composition des couches et la couche "active" est une étape importante dans la conception d'IVP à la fois performant et plaisant pour l'utilisateur.

CHAPITRE 5

Interaction Rythmique

5.1 Introduction

L'intégration de nouvelles techniques d'interaction dans un système existant est un problème récurrent. En particulier, l'incorporation de fonctionnalités innovantes dans les gestionnaires de fenêtres est compliquée par le rôle de ce système. Son but est entre autres d'offrir ses services aux applications et de ne pas en gêner l'utilisation. Or, les applications sont souvent complexes, et les méthodes d'entrée traditionnelles déjà largement employées par les interactions nécessaires au fonctionnement de celles-ci. Deux solutions s'offrent alors aux concepteurs pour intégrer de nouvelles techniques dans un gestionnaire de fenêtres. Soit ils ignorent tout simplement le problème et choisissent des interactions simples, mais risquant d'interférer avec les fonctionnalités de certaines applications. Soit ils contournent cette difficulté en choisissant par exemple des raccourcis clavier compliqués à retenir et à effectuer, ou bien en plaçant un élément graphique comme un bouton permettant l'activation de cette fonctionnalité dans une partie de l'interface (barre de titre des fenêtres d'applications, fenêtre supplémentaire dédiée, etc.). Malheureusement, aucune de ces solutions n'est satisfaisante car elles peuvent réduire fortement l'utilité pratique d'une nouvelle fonctionnalité.

Dans le chapitre 2 de cette thèse, nous avons présenté un moyen d'intégrer les historiques de copier-ou-déplacer (section 2.2) en séparant de manière claire les interactions destinées aux applications (les opérations classiques de copier-ou-déplacer) de celles destinées au gestionnaire de fenêtres (les opérations augmentées par les historiques), et en conservant l'interaction avec le système localisée autour du point d'intérêt de l'utilisateur. Pour ce faire nous avons utilisé des interactions temporisées (voir la Figure 2.2). Nous aurions pu nous contenter d'évaluer ces interactions, mais l'utilisation du temps dans l'interaction est apparue comme un sujet plus riche que l'exploitation de simples délais. Cela nous a conduit à travailler avec Emilien Ghomi, doctorant de l'équipe In Situ, sur une notion plus générale : l'interaction rythmique. Les travaux rapportés dans la suite de ce chapitre ont été effectués en commun.

En interaction homme-machine, peu d'études se sont penché sur les possibilités d'utilisation du temps comme une dimension contrôlable et pouvant être maitrisée par l'utilisateur afin d'améliorer l'interaction avec les systèmes interactifs. Toutefois, certaines techniques utilisant des structures temporelles existent depuis longtemps : clics multiples et interactions temporisées ("spring loaded floders" du Macintosh) en sont de bons exemples. Cependant, l'interaction temporelle, et plus particulièrement l'utilisation de rythmes, n'a pas été étudiée de manière systématique.

Nous présentons ici une première étude de cet espace de conception, que nous nommons "*Interaction Rythmique*", par le biais de deux expériences utilisateurs. L'interaction rythmique, à l'inverse de la plupart des interactions proposées à l'heure actuelle qui utilisent la dimension spatiale (association de positions pour les raccourcis clavier, mouvements pour les interactions gestuelles), considère la dimension temporelle. Nous en proposons une concrétisation possible sous la forme de règles définissant une grammaire de *motifs rythmiques* ainsi qu'un ensemble de reconnaisseurs permettant à un système interactif d'utiliser l'interaction rythmique. Nous proposons par la suite un vocabulaire et explorons les possibilités de reproduction de ces motifs par les utilisateurs ainsi que les capacités de mémorisation de leur association avec des commandes.

5.2 Motivations

Le Rythme est un élément omniprésent dans notre vie quotidienne. C'est l'une des composantes principales de la prosodie nécessaire à la complète compréhension du langage. Il nous aide également à percevoir la dimension invisible et continue qu'est le temps grâce, par exemple, au *tic-tac* d'une pendule. Il peut même être le fondement de certaines structures sociales. Par exemple, la tribu africaine *Ewe* possède une culture et une organisation sociale basées sur l'habileté des individus aux percussions. Cette capacité humaine à percevoir et à interpréter les rythmes est considérée comme une des plus importantes par les psychologues, qu'elle soit utilisée afin d'interagir avec notre environnement ou même comme thérapie (dans [99], Sacks parle de l'utilisation de pulsations rythmique pour aider les personnes atteintes de la maladie de Parkinson à récupérer leur mobilité). Cependant, bien que la perception et la reproduction de rythmes soient des capacités fondamentales de l'être humain, elles sont assez rarement utilisées comme vecteur d'interaction avec l'outil informatique.

L'interaction avec les environnements de bureau traditionnels repose encore essentiellement sur la manipulation d'objets graphiques à l'aide de la souris (boutons, menus, icônes...) et sur l'usage de raccourcis clavier pour déclencher certaines commandes. Plusieurs alternatives ou améliorations de ce paradigme existent. Par

exemple des gestes continus exécutés avec un dispositif de pointage ou les clics multiples sur les boutons de la souris permettent d'enrichir le vocabulaire d'actions en introduisant les dimensions *spatiales* et *temporelle*. Toutefois, alors que la dimension spatiale a retenu l'attention des chercheurs en interaction homme-machine, engendrant par exemple des interactions gestuelles ou basées sur les postures des mains [6, 9, 72], la dimension temporelle a été peu étudiée. De plus, certaines de ces techniques requièrent de la part de l'utilisateur l'exécution d'actions complexes, de prime abord difficiles à comprendre et à effectuer. Cependant, cette complexité peut aussi être un avantage, offrant une aide à la mémorisation motrice puisque plusieurs niveaux d'encodage permettent d'améliorer la mémorisation [30]. De la même façon, nous pouvons espérer que l'introduction d'informations temporelles et auditives dans un motif rythmique puisse aider l'utilisateur à mémoriser (encoder) une association entre une commande et un motif rythmique de la même façon que dessiner une forme dans le cas de l'interaction gestuelle.

5.2.1 Avantages du Rythme comme Méthode d'Entrée

Nous avons identifié un certain nombre d'avantages potentiels à l'utilisation du rythme pour l'interaction avec un système interactif. Tout d'abord, comme démontré par les sciences cognitives, il existe une correspondance directe entre jouer un rythme (action) et écouter un rythme (les possibles stimulus et feedback audio). Ensuite, les rythmes peuvent être joués dans de nombreuses situations. Jouer un rythme ne requiert guère plus qu'un degré de liberté et une grande variété de mouvements peuvent être réalisés en rythme, comme par exemple taper du doigt, taper des pieds ou hocher la tête, et être enregistrés par différents capteurs.

L'interaction gestuelle requiert généralement un certain espace pour être réalisée. La réalisation de ces gestes sur de petits écrans tactiles interfère d'ailleurs souvent avec l'espace d'affichage. Au contraire, les rythmes peuvent être joués sur une petite surface de l'écran tactile et peuvent même être exécutés sans avoir à regarder l'appareil.

Enfin, les structures rythmiques peuvent être conçues de manière hiérarchique. L'utilisation de préfixes communs pour différents motifs peut conduire à une hiérarchie naturelle qui pourrait être acquise par les utilisateurs, renforçant la mémorisation du vocabulaire d'interaction.

5.2.2 Utilisation des Motifs Rythmiques

Les motifs rythmiques n'ont pas pour vocation de remplacer les techniques de déclenchement de commandes conventionnels. Au contraire, il s'agit d'une alternative pouvant se révéler plus adaptée dans des situations spécifiques comme celle

d'une interaction avec un appareil sans avoir à le regarder. C'est également une opportunité pour augmenter les méthodes d'entrée actuelles afin de définir un vocabulaire d'interaction plus riche. Par exemple, les motifs rythmiques pourraient permettre l'accès à un petit nombre de commandes comme la numérotation rapide d'un numéro de téléphone pré-enregistré. Ils pourraient aussi permettre la navigation dans un livre électronique ou le changement de mode dans une application.

Dans d'autres situations, les motifs rythmiques pourraient simplifier l'interaction. Par exemple, les marques-pages, les menus et les fiches de contacts sont souvent organisés de manière hiérarchique. La structure des motifs rythmiques pourrait imiter cette hiérarchie ou au contraire proposer une hiérarchie alternative, comme par exemple classer les contacts par prénom. De plus, comme jouer des motifs rythmiques ne mobilise pas la vision de l'utilisateur, l'interaction rythmique est indiquée pour l'utilisation d'un dispositif tactile dans la poche ou pendant que l'utilisateur conduit une voiture. Les motifs rythmiques pourraient même être utilisés dans le noir pour par exemple éteindre l'alarme d'un réveil. Leur utilisation est même envisageable dans le cas de dispositifs ne possédant aucun affichage.

Finalement, l'Interaction Rythmique offre également des pistes nouvelles pour répondre à des problèmes connus. L'exécution d'un rythme sur le dos d'un dispositif mobile tenu dans la main peut être capturée sans l'ajout de capteurs supplémentaires grâce aux accéléromètres que l'on trouve sur la plupart des smartphones et des tablettes. Par exemple, un motif rythmique joué pendant un appel téléphonique permettrait d'ajouter le correspondant à la liste des contacts. Une telle interaction pourrait également afficher des informations supplémentaires telles que le niveau de la batterie ou du signal. De plus, ces motifs pourraient être joués avec la main non dominante voire même avec une autre partie du corps comme taper des pieds [102] pour refuser un appel.

Cependant, bien que nous considérions l'utilisation de rythmes comme prometteuse, l'interaction rythmique n'en est encore qu'à ses balbutiements. Avant de pouvoir l'étudier plus amplement et de concevoir des techniques d'interaction concrètes tirant parti de ses avantages, des questions fondamentales doivent être étudiées :

- *Faisabilité*. Bien que la perception et la reproduction de rythmes soient quasi naturelles pour les êtres humains, est-il viable d'introduire ce moyen de communication dans un système interactif ? Les utilisateurs seront-ils capables de reproduire assez fidèlement les motifs pour être "compris" par le système ? Seront-ils capables d'apprendre ces mêmes motifs et de retenir leur association avec des commandes du système ?
- *Conception de l'interaction*. Le nombre de motifs rythmiques est infini et leur mode de présentation à l'utilisateur peut également varier. Quels sont les motifs les plus appropriés pour l'interaction avec un ordinateur ? Quel

type de feedback est efficace lorsque l'utilisateur reproduit le motif ou pour aider à l'apprentissage ?
– *Considérations techniques & Intégration*. Comme toute technique d'interaction de haut niveau telle que la reconnaissance vocale, les marques ou les gestes, l'interaction rythmique se base sur un "reconnaisseur" qui segmente et interprète les actions de l'utilisateur. Quels sont les besoins et les contraintes d'un tel reconnaisseur ? De plus, comment cette technique et le retour lié à l'interaction rythmique pourraient-ils être intégrés dans des systèmes interactifs existants ?

5.3 État de l'art

Il existe une littérature abondante en sciences cognitives et en neuropsychologie traitant de la perception, de la reproduction et de l'utilisation du rythme avec des objectifs et des points de vue différents : par exemple, la psychologie a exploré la perception et l'action [44, 74] tandis que la linguistique appliquée et artistique ont été appliquées à la musique en lien avec l'apprentissage [90, 83]. Pour une revue détaillé des travaux en psychologie portant sur les rythmes nous reportons le lecteur à [42] et [28]. Dans cette section nous nous concentrons sur le rythme comme méthode d'entrée en interaction homme-machine.

Le rythme repose sur le temps, qui est lui-même une dimension parfois utilisée en interaction, essentiellement comme moyen de distinguer les actions. Par exemple, certaines interfaces graphiques font la différence entre un clic court ou long, chacun d'eux déclenchant une commande différente. Ce principe de temporisation, c'est-à-dire suspendre l'interaction pendant un temps donné, est également utilisé comme moyen pour segmenter les actions dans l'interaction gestuelle [54]. Nous avons également utilisé ce principe dans le chapitre 2 pour désigner explicitement le destinataire d'une commande en déterminant que les actions "courtes" avaient pour valeur les actions usuelles et étaient destinées à l'application courante alors que les "longues" étaient les actions augmentées d'historique et avaient pour cible le système.

Cliquer plusieurs fois sur les buttons de la souris de manière rapide (double clics, triple clics, clic et demi) est certainement la plus simple et la plus utilisée des interactions rythmiques. Malheureusement, la périodicité et le rythme n'ont été que rarement étudiés en tant que dimension possible pour l'interaction. Un des premières travaux exploitant le temps en tant que dimension d'interaction est les Rhythmic Menus [79]. Ces travaux présentent des menus linéaires dont l'élément sélectionné change à une fréquence donnée tant que l'utilisateur appuie sur un bouton. Lorsqu'il relâche le bouton, l'élément actuellement sélectionné est activé. De façon similaire, Motion Pointing [38] utilise des motifs elliptiques périodiques

associés à des objets permettant de sélectionner un objet particulier en effectuant un mouvement oscillatoire en phase avec le motif associé à l'objet désiré.

À l'opposé de ces approches orientées système, la technique Cyclostar [77] permet à l'utilisateur de contrôler des paramètres continus comme la vitesse de zoom, en effectuant des mouvements oscillatoires cycliques. Cependant dans tous ces cas, l'aspect rythmique est réduit à sa plus simple expression et des situations simples dans lesquelles le mouvement est périodique et équivalent pour chaque période.

Seule une poignée de technique fait appel à la reproduction de rythmes complexe. Deux exemples notables sont Five-key [113] et Tapsongs [119]. Five-key est un système d'entrée de texte se servant de séquences rythmiques. Les caractères sont entrés en effectuant une série d'appuis "longs" ou "courts" sur des touches à l'aide de deux doigts uniquement. L'avantage est de réduire le mouvement des doigts en utilisant seulement cinq touches. Cependant, l'efficacité de cette méthode et l'apprentissage indispensable à son utilisation n'ont pas été étudiés en profondeur. Tapsongs pour sa part est une approche rythmique des mots de passe remplaçant l'usage de texte par celui de la reproduction de rythmes. Tapsongs tente de différencier les utilisateurs par leur façon de taper des rythmes. Notre but est donc à l'opposé de Tapsongs puisque nous cherchons à créer un reconnaisseur capable de trouver le motif rythmique entré indépendamment des différences entre les utilisateurs et sans entraînement préalable du reconnaisseur.

Quelques travaux ont considéré des motifs rythmiques en utilisant d'autres moyens d'entrée que le fait de taper avec son doigt ou la périodicité des gestes. Par exemple un accéléromètre est utilisé par Lantz et Murray-Smith [73] pour étudier le rythme des gestes avec un dispositif mobile. Westeyn et Starner [118], quant à eux, regroupent les clignements des yeux en motifs rythmiques afin de les utiliser comme méthode d'entrée pour des personnes handicapées. Cependant dans ce dernier cas, le reconnaisseur nécessite un entrainement important et la durée des clignements n'est pas prise en compte.

5.4 Des Motifs Rythmiques pour l'Interaction

Nous présentons dans cette section ce que nous appellons par la suite "motifs rythmiques" qui seront "joués" sur un trackpad dans les expériences utilisateur. Notre définition des motifs rythmiques est fortement inspirée de la musique et en particulier de la "phrase rythmique". En musique, la structure élémentaire est appelée *phrase* ou *passage*. Elle est définie comme une "cellule mélodique, rythmique ou harmonique" ([78] page 2052). La *Phrase Rythmique* est indépendante des aspects mélodiques ou harmoniques et elle représente la structure temporelle d'un

ensemble de notes. Elle consiste en la durée relative des notes et des silences existant entre celles-ci. La hauteur de note n'importe pas et la phrase rythmique peut être interprétée à n'importe quel tempo tant que l'interprète conserve les durées relatives des notes. L'articulation (continuité, recouvrement ou séparation entre les notes) et les accents (modulation en intensité de certaines notes) peuvent faire partie du motif rythmique. Dans la musique occidentale moderne, notes et silences peuvent prendre 8 durées différentes (il est possible d'accoler autant de notes et de silence que désiré), et les motifs peuvent contenir autant de notes que désiré permettant un très grand nombre de motifs rythmiques possibles.

En tenant compte du nombre de commandes et d'actions possibles lors de l'utilisation d'un système interactif, une telle puissance d'expression n'est pas nécessaire en interaction homme-machine. C'est pourquoi nous proposons une version limitée des phrases rythmiques, plus adaptée à l'interaction homme-machine, que nous nommons motif rythmique (ou simplement "motif"). Un motif rythmique est un regroupement d'événements (pleins) et de silences (vides) dont les durées sont mesurées en pulsations. Ils sont définis par les règles suivantes introduites pour réduire la complexité des motifs rythmiques :

- Les événements peuvent être de trois types différents : **impulsion**, **court** ou **long** ;
- les événement courts et longs commencent au début d'une pulsation et durent respectivement 1 et 2 pulsations ;
- l'impulsion est l'événement le plus court. Il correspond à un contact très court avec le dispositif d'entrée. Comme les autres événements, il commence au début d'une pulsation mais ne dure qu'une fraction de celle-ci.
- les silences peuvent être "court" ou "long" et possèdent les même durées que les événement correspondants (1 et 2 pulsations).
- Pour des raisons évidente de segmentation de l'interaction, les motifs doivent commencer et finir par un événement.
- Deux silences ne peuvent se suivre.

La **taille** d'un motif est la somme des durées des évènements et des silences la composant. Dans ce chapitre nous nous limitons à l'étude de motifs dont la taille est comprise entre 2 et 6 pulsations pour des raisons de reproduction et de mémorisation. Le nombre de motifs qu'il est possible de générer en suivant ces contraintes est le suivant :

- 5 motifs à 2 pulsations ;
- 16 motifs à 3 pulsations (voir la Figure 5.1) ;
- 53 motifs à 4 pulsations ;
- 171 motifs à 5 pulsations ;
- 554 motifs à 6 pulsations ;

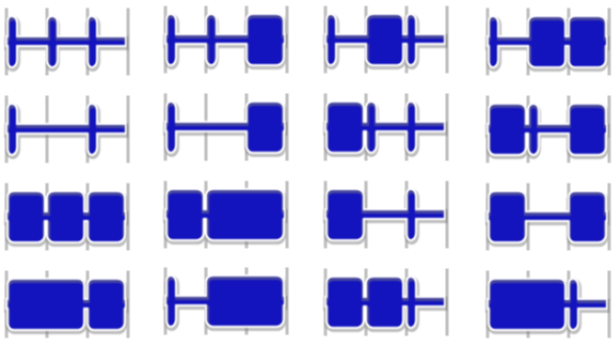


FIGURE 5.1 – *Les seize motifs de taille trois pulsations que l'on peut construire en suivant nos contraintes. Chaque rectangle représente un événement. Les fines lignes grises indiquent le début d'une pulsation.*

Le nombre total de motifs à n événements peut être calculé grâce à la formule suivante : 3^{2n-1}, soit 199.290 motifs de 6 événements ou moins.

Dans le reste de cette étude nous considérons un tempo universel de 120 pulsations par minute (i.e. 2 Hz). De ce fait les débuts de deux événements sont séparés d'au moins 500 ms. Cette valeur est considérée comme la fréquence préférentielle du système moteur de l'être humain, par exemple pour la marche [74], et le tempo de prédilection en musique [83].

Au cours de cette étude préliminaire nous ne considérerons comme dispositif d'entrée qu'une surface tactile (touchpad). Bien que l'utilisation d'un clavier, des boutons d'une souris, d'accéléromètres [73] ou du clignement des yeux [118] sont autant de possibilités envisageables, nous les considérons hors du champs de cette étude.

L'une des tâches importantes de cette recherche est de concevoir un reconaisseur fiable, capable d'identifier les motifs joués par les utilisateurs. Dans la première expérience, nous avons utilisé un reconaisseur *structurel* pour s'assurer des capacités de reproduction des motifs par des utilisateurs novices. En prenant en considération les résultats de cette première expérience, nous avons conçu un *classificateur de motifs*, moins strict, permettant aux utilisateurs quelques maladresses tout en discriminant les motifs d'un vocabulaire restreint. Ce classificateur a été utilisé lors de la seconde expérience qui teste la capacité des utilisateurs à mémoriser l'association d'un motif rythmique à une commande dans un contexte applicatif.

5.5 Expérience 1 : Reproduction de Motifs Rythmiques

Nous avons mené une expérience afin d'évaluer la capacité d'utilisateurs novices à reproduire des motifs rythmiques avec assez de précision pour permettre la création d'interactions utilisant ce mode d'entrée. De plus, nous avons comparé les effets de différents feedbacks lorsque les utilisateurs jouent les motifs demandés.

5.5.1 Le Reconaisseur

Le reconnaisseur que nous avons conçu pour cette première expérience se base sur notre définition des motifs rythmiques (i.e., types d'événements ou de silences et durées relatives) mais reste indépendant du vocabulaire considéré. Il commence par extraire la structure rythmique de la séquence d'entrée. Cette structure est une liste d'évènements et de silences dont la durée relative a été déterminée (impulsion, court, long) par des heuristiques.

Nous utilisons l'algorithme de partitionnement de données des *k-means* [66] itéré 500 fois sur la durée des tapes de la séquence d'entrée pour déterminer les durées relatives de chaque événement et silences. L'algorithme regroupe les tapes puis essaie de déterminer la nature probable de chaque partition : impulsions, courts ou longs. Notre algorithme s'assure d'une distance minimale de 200 *ms* entre chaque partition. Si deux partitions sont séparées par un temps de moins de 200 *ms* elles sont combinées pour ne former plus qu'une seule partition et leurs éléments seront reconnus comme étant des représentants du même type d'événement. Cette limite correspond à un tempo minimal pour que les motifs rythmiques puissent être reconnus. C'est pourquoi, si les motifs sont joués trop rapidement, notre reconnaisseur pourrait confondre des événements de types différents. Les valeurs de référence utilisées pour l'identification des partitions sont : 1000 *ms* pour les événements ou les silences longs, 500 *ms* pour les courts et 180 *ms* pour les impulsions et les relâchements (le pendant des impulsions pour les silences). Ces valeurs correspondent évidemment aux valeurs absolues pour les types d'événements et de silence considérés, joués à un tempo de 120 pulsations par minutes.

Après le partitionnement et la détermination du type des événements, les silences succédant une impulsion sont éliminés du motif reconstitué (voir la figure 5.2). Finalement, ce motif est comparé à ceux du vocabulaire afin de savoir s'il correspond bien à celui utilisé pour le stimulus donnée au participant.

Avec des connaissances minimales sur notre concept de motifs rythmique, cet algorithme peut déterminer le type de chaque évènement et de chaque silence de la séquence d'entrée et cela même dans des situations défavorables comme l'occurrence d'un unique type d'événement dans le motif demandé. Grâce au partitionnement des évènements et des silences, ce reconnaisseur est peu sensible aux motifs

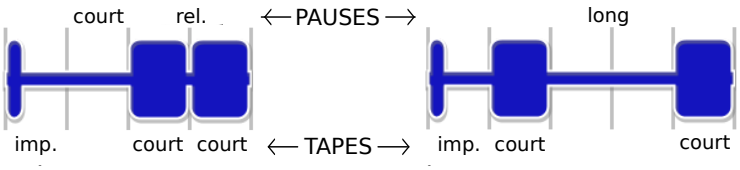


FIGURE 5.2 – *Ces deux motifs rythmiques entrainent une même séquence de tapes : une impulsion suivie de deux événements courts. Cependant, ils diffèrent dans le placement et la durée des silences et possèdent donc une structure rythmique différente. Dans cette figure,* rel. *signifie relâchement (l'équivalent de l'impulsion pour les silences) et* imp. *impulsion.*

homothétiques et s'adapte (dans une certaine mesure) au tempo choisi par le participant. Le tempo d'une séquence d'entrée peut d'ailleurs être calculé en comparant les centroïdes des partitions.

5.5.2 Matériel et Participants

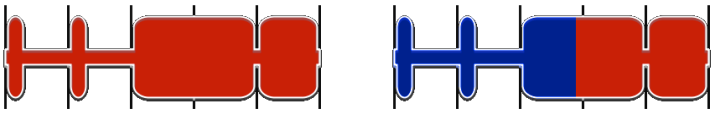
L'expérience a été réalisée en Java avec l'aide de TouchStone [75] et conduite sur un portable Apple MacBook (Processeur Intel). Les motifs rythmiques étaient joués sur le touchpad embarqué de la machine. L'utilisation d'un tel dispositif permet de supprimer les contraintes mécanique dues au dispositif (en pratique les rythmes pourraient être joués en cliquant sur le buttons d'une souris, en tapant sur les touches d'un clavier ou en tapant sur un écran tactile).

Douze participants bénévoles dont six femmes ont participé à cette expérience, âgés de 23 à 53 ans avec une moyenne de 29 ans et une médiane à 27 ans. Parmi ces douze participants, cinq n'avaient jamais pratiqué la musique.

5.5.3 Stimulus

Le motif rythmique dont la reproduction est demandée est présenté au participant de la manière la plus complète possible. Nous avons pour cela combiné les modalités visuelles et auditives. La représentation graphique utilise conjointement une représentation statique et anime le remplissage de cette représentation de façon synchrone avec le stimulus audio.

Lors de son apparition, le stimulus n'est constitué que d'une forme statique représentant le motif dans son ensemble. Dans cette forme, chaque rectangle représente un événement (Figure 5.3.a). Puis cette forme est remplie progressivement



(a) **(b)**

FIGURE 5.3 – *Le stimulus utilisé lors de la première expérience. Une représentation statique est d'abord affichée* ***(a)****. Puis elle est remplie de manière synchrone avec la représentation audio du motif* ***(b)****.*

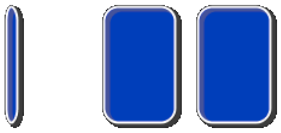


FIGURE 5.4 – *Feedback visuel lors de la reproduction d'un motif par l'utilisateur pour l'expérience de reproduction de motifs rythmiques.*

(Figure 5.3.b) en synchronisation avec le stimulus audio. Pour faciliter l'interprétation de la durée des événements, une fine ligne grise marque le début de chaque pulsation.

Le stimulus audio utilise une hauteur de note de 440Hz qui correspond à la note "La" servant pour accorder les instruments de musique (ISO 16 :1975 [1]). Cette note est jouée par l'instrument MIDI nommé "English Horn" à un niveau sonore constant. Le choix de cet instrument a été motivé par sa douceur permettant aux participants de supporter ce son tout au long de l'expérience, mais il dispose également d'une attaque et d'une fin claires et facilement perceptibles.

5.5.4 Feedbacks Utilisateur

Lors de la reproduction des motifs rythmique par les participants, nous avons considéré quatre feedbacks (FEEDBACK).

Le feedback $Audio$ utilise le même son que pour le stimulus lorsque le doigt du participant est en contact avec la surface tactile. Le feedback $Visual$ est basé sur la représentation graphique utilisée pour le stimulus : les rectangles représentant les événements sont affichés dynamiquement pendant que le participant tape les évènements sur la surface tactile (voir la Figure 5.4). La condition $AudioVisual$ combine les deux présentations audio et visuel. Enfin, dans la condition $None$, aucun feedback n'est fourni au participant.

1. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=3601

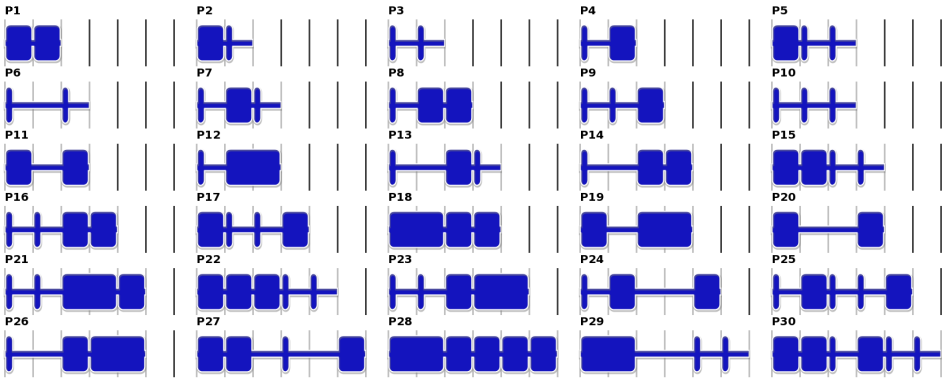


FIGURE 5.5 – *Le vocabulaire utilisé lors de l'expérience de reproduction des motifs rythmiques.*

Les trois feedbacks possibles : $Audio$, $Visual$ et $AudioVisual$ pourraient être utiles lors de phases d'apprentissage de reproduction de motifs rythmiques, alors que la dernière condition $None$ pourrait avoir son utilité lorsque l'utilisateur doit enter une commande sans regarder dans un environnement bruyant ou silencieux.

5.5.5 Vocabulaire

Pour cette expérience, nous avons choisi 30 motifs rythmiques parmi les 799 qu'il est possibles de générer entre 2 et 6 pulsations en suivant les règles exposées dans la section précédente. Ces motifs rythmiques sont présentés dans la Figure 5.5. Ce vocabulaire est composé de :

- 4 motifs de 2 pulsations ;
- 8 motifs de 3 pulsations ;
- 8 motifs de 4 pulsations ;
- 6 motifs de 5 pulsations ;
- 4 motifs de 6 pulsations.

Ce qui nous donne un peu moins de motifs pour les conditions extrêmes que nous testons. Pour chaque longeur en pulsations, nous avons choisi d'éviter de déséquilibrer les nombres d'événements. Par exemple, pour les motifs de 4 pulsations :

- 2 sont composés de 2 événements ;
- 3 sont composés de 3 événenemts ;
- 3 sont composés de 4 événements.

5.5.6 La Tâche

Lors de cette expérience, un essai se déroule de la façon suivante. Le motif à reproduire est joué deux fois de suite au participant en utilisant le stimulus (voir section 5.5.3). Les sujets doivent ensuite reproduire le motif demandé le plus précisément possible à l'aide du majeur de leur main dominante en tapant sur le pavé tactile.

Après que le reconnaisseur ait calculé la structure temporelle de la séquence entrée par le participant, le système indique à ce dernier si sa performance est reconnue comme étant le motif demandé.

5.5.7 Procédure et Schéma Expérimental

Cette expérience suit un schéma intra-sujet 4×30 avec les facteurs suivants :

- FEEDBACK : $Audio$, $Visual$, $AudioVisual$ and $None$;
- PATTERN : P1 – P30 (Figure 5.5).

Au début de la session expérimentale, chacune des conditions de FEEDBACK est présentée au participant par de courts blocs de 15 essais avec des motifs pris au hasard. Puis le sujet doit réaliser deux blocs d'entrainements de 15 essais chacun dans la condition que nous considérons comme la plus susceptible de guider les participant dans leur tâche de reproduction : $AudioVisual$. Ces deux blocs doivent permettre aux participants de s'habituer à la tâche qui leur est demandée.

Pendant la session principale, les essais sont regroupés en blocs selon la condition FEEDBACK. L'ordre d'apparition de cette dernière condition est contre-balancé entre les participants par l'utilisation d'un carré latin. Dans chaque bloc les 30 motifs sont utilisés deux fois avec un ordre d'apparition aléatoire. Avant chaque bloc faisant l'objet de mesures, un bloc d'entraînement de 15 essais est présenté au sujet pour lui rappeler la condition. Les participants peuvent se reposer entre et au milieu de chaque bloc (tous les 15 essais).

Nous avons donc recueilli 2880 mesures = 12 (participants) $\times$ 4 (FEEDBACK) $\times$ 30 (PATTERN) $\times$ 2 (répétitions). Nous avions demandé aux participants d'être le plus précis possible et de bien différentier la durée des trois types d'événements et de silences. Chaque participant a complété sa session en 1 heure environ. Il leur était alors demandé de classer les types de feedback utilisateur par la difficulté éprouvée à effectuer la tâche de reproduction sur une échelle de Likert à 5 échelons.

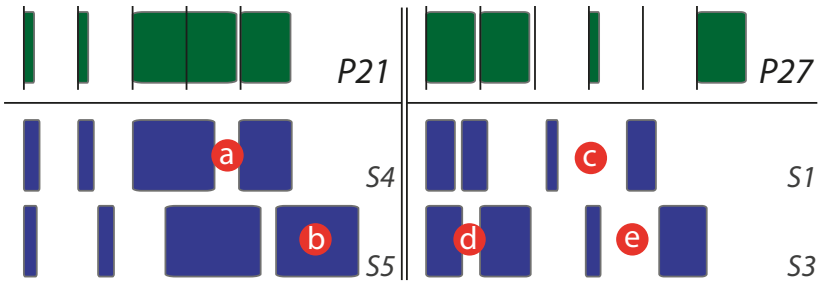


FIGURE 5.6 – *Deux stimuli (21 et 27) comportant des erreurs de reproduction dans l'interprétation faite par des participants à l'expérience 1. S4 : la dernière pause est trop longue (a) ; S5 : la dernière tape est trop longue (b) ; S1 : la dernière pause est trop courte (c) ; S3 : la première pause est trop longue (d) et la dernière est trop courte (e).*

5.5.8 Résultats Quantitatifs

Le taux de reconnaissance globale est de 64.3%. Ce score peut paraître faible, mais il faut garder à l'esprit que notre reconnaisseur est volontairement très strict sur la structure temporelle des motifs. Il peut reconnaître les 799 motifs possibles comportant entre deux et six pulsations et pas seulement les 30 motifs utilisés dans cette étude. La reproduction précise des motifs rythmiques demandés dans cette expérience s'apparente à jouer des percussions. Bien que ce soit une tâche familière pour des musiciens, il faut généralement des années pour maîtriser ces compétences.

La figure 5.6 montre des exemples représentatifs d'erreurs de reproduction de la part des participants à cette étude, comme des relâchements trop longs qui sont reconnus comme des pauses courtes ou des pauses et des tapes dont la durée a été mal interprétée et qui sont donc mal reconnues lors de la phase de partitionnement. Il est intéressant de noter que les erreurs sont plus fréquentes sur les pauses que sur les tapes, ce qui est en accord avec les travaux de Rammsayer [92] montrant que les utilisateurs sont plus précis à reproduire une note qu'un silence.

Une ANOVA pour le facteur FEEDBACK (avec les participants comme variable aléatoire) montre un effet significatif sur le taux de succès ($F_{3,33} = 15.4$, $p < 0.0001$). Cet effet peut être observé sur la figure 5.7 [2]. Une analyse post-hoc utilisant un t-test avec correction de Bonferroni indique que la condition $None$ est significativement moins bonne que toutes les autres conditions de feedback. Il n'est pas surprenant que l'absence de feedback utilisateur lorsque le participant joue le motif dégrade la précision de la reproduction.

2. Les barres d'erreur de chaque figure montrent l'intervalle de confiance à 95%.

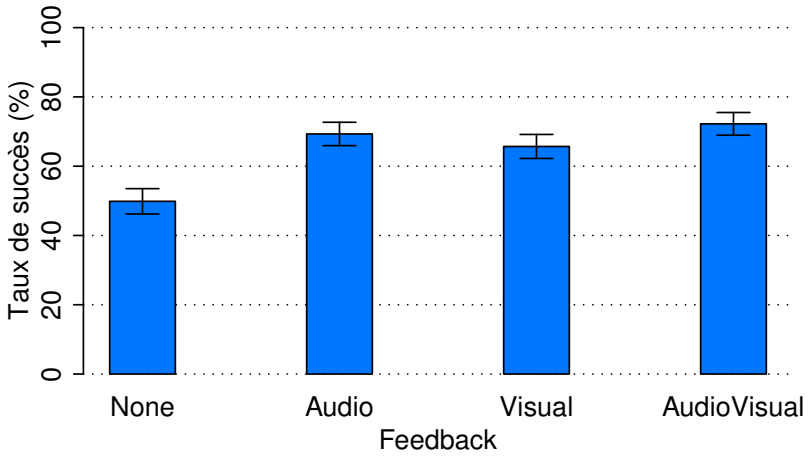


FIGURE 5.7 – *Taux de succès pour chaque* FEEDBACK.

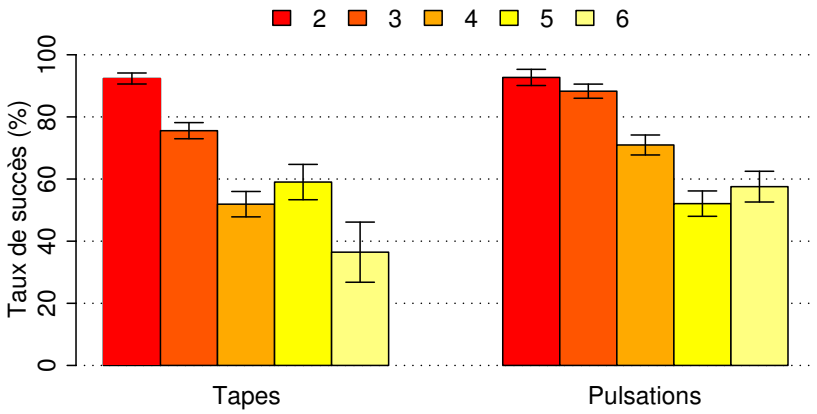


FIGURE 5.8 – *Taux de succès en fonction du nombre d'événements et de la longueur en pulsations des motifs.*

En ce qui concerne le vocabulaire (PATTERN), une ANOVA montre un effet significatif sur le taux de succès ($F_{29,319} = 18.2$, $p < 0.0001$). Nous observons de grandes variations du taux de succès entre les motifs, allant de 16% pour P27 à 98% pour P10. Une quinzaine de motifs ont un taux de reconnaissance d'au moins 70% : P1–P7, P9–P13, P19, P20, and P29. Tous ces motifs ont au moins 3 événements et, mis à part P29 tous ont moins de 4 pulsations. Au contraire, quelques motifs à trois événements ont un taux de reconnaissance faible (en dessous de 50%) : P14 et P18 (tous deux ayant 4 pulsations). Un t-test post-hoc avec une correction de Bonferroni ou de Holm [3] révèle un grand nombre de paires de motifs avec des différences significatives.

3. La correction de Holm est plus adaptée (et moins stricte) que la correction de Bonferroni en présence d'un grand nombre de valeurs.

Il est difficile de dégager des propriétés des motifs qui impliquent une difficulté de reproduction. Toutefois, le nombre d'évènements et la longueur (en pulsation) sont les caractéristiques les plus évidentes d'une certaine complexité d'un motif. En effet, le nombre d'évènements (i.e., de tapes à produire) et la longueur d'un motif ont un effet significatif sur le taux de succès (tapes : $F_{4,44} = 54.1, p < 0.0001$, longueur : $F_{4,44} = 85.5, p < 0.0001$), sans interaction significative avec FEEDBACK. Des t-test post-hoc avec correction de Bonferroni, montrent en effet que dans la plupart des cas le meilleur taux de succès a été obtenu avec des motifs ayant peu d'évènements et/ou peu de pulsations (voir la Figure 5.8).

Cherchant à analyser la complexité des motifs rythmiques, nous avons également étudié le nombre d'inversions dans un motif. Cette caractéristique est dérivée de la durée relative des événements d'un motif. On considère qu'au début d'un motif, il n'y a aucune inversion et que les durées des événements doivent croitre. En considérant chaque événement du motif dans l'ordre, si une durée est plus courte que la précédente, alors il y a une inversion et les durées sont maintenant considérées comme décroissantes. L'algorithme continue ainsi jusqu'à la fin du motif en incrémentant le nombre d'inversions chaque fois qu'un changement dans la progression des durées est observé.

Une ANOVA sur ce facteur démontre un effet significatif sur le taux de succès ($F_{3,33} = 19.7, p < 0.0001$), aucune interaction étant détectée avec FEEDBACK. Une analyse a posteriori utilisant un t-test avec correction de Bonferroni dévoile une différence significative pour tous les niveaux de NINV sauf entre 2 et 3 (voir la Figure 5.9).

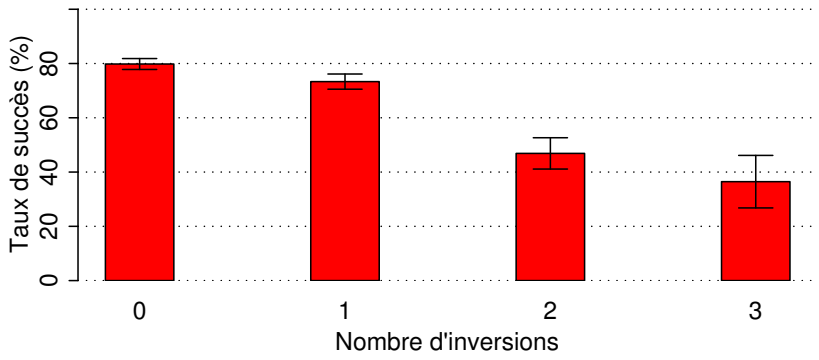


FIGURE 5.9 – *Taux de succès en fonction de* NINV.

L'analyse présentée ci-dessus montre que, sans réelle surprise, le feedback reste un facteur important afin d'obtenir une meilleure reproduction des motifs rythmiques par les utilisateurs. Cependant, aucun des feedbacks testés dans cette expérience ne semble plus approprié en soi. Le feedback pourra et devra donc être choisi en fonction des situations. En ce qui concerne la difficulté de reproduction des motifs rythmiques en eux-mêmes, plus ceux-ci sont longs ou plus ils possèdent

de tapes, plus leur reproduction est ardue. Cependant ces deux facteurs ne suffisent pas à expliquer toutes les différences constatées en pratique dans les données récoltées au cours de cette expérience, et nous avons donc recherché d'autres facteurs pouvant influer la complexité perçue d'un motif. Le nombre d'inversions d'un motif semble être l'un de ces facteurs. La complexité d'un motif reste cependant un facteur difficile à évaluer et dépendant de beaucoup de variables. Nous fournissons dans cette analyse un début d'étude de cette complexité, sans toutefois être en mesure de donner une réponse définitive à la question "Pourquoi ce motif est-il plus difficile à reproduire que celui-ci ?". Finalement, l'utilisation d'un reconnaisseur structurel strict nous a appris que les participants sont moins précis dans la reproduction de la longueur d'un silence que d'une tape.

5.5.9 Résultats Qualitatifs

Six des 12 participants ont préféré le feedback $Audio$, 3 ont préféré $Visual$, et 2 ont préféré $AudioVisual$. Enfin 1 seul à préféré l'absence de feedback ($None$). De plus, 6 participants ont classé $AudioVisual$ second et 8 ont classé $None$ dernier. De nombreux participants ont noté que le feedback $AudioVisual$ était déroutant, car il donnait trop d'informations simultanées. Ils ont expliqué que la plupart du temps, ils choisissaient une des composantes du feedback (audio ou visuel) et essayaient d'ignorer l'autre.

Concernant la difficulté subjective de la tâche, 7 des participants n'étaient pas d'accord avec l'affirmation "J'ai trouvé difficile de reproduire les motifs rythmiques", alors que 4 n'avaient pas d'avis, ne laissant qu'un participant d'accord avec cette affirmation. Ce dernier participant n'était toutefois pas d'accord avec l'affirmation dans le cas des feedbacks $None$ et $Visual$.

L'avis des participants sur l'interaction rythmique est encourageant, surtout si l'on considère que les résultats de succès ou d'échec présentés aux sujets lors de l'expérience utilisaient un reconnaisseur strict, rendant la tâche plus difficile et probablement quelque peut frustrante pour les participants dans le cas d'erreurs consécutives. Notons que les résultats qualitatifs sont en accord avec les résultats quantitatifs concernant l'utilité des feedbacks.

5.6 Outil d'Analyse : AnEwe

L'analyse des entrées des participants à la première expérience présentée dans la section précédente est une tâche complexe. Si l'analyse statistique présentée dans les résultats des expériences donne de précieuses informations, tous les détails de

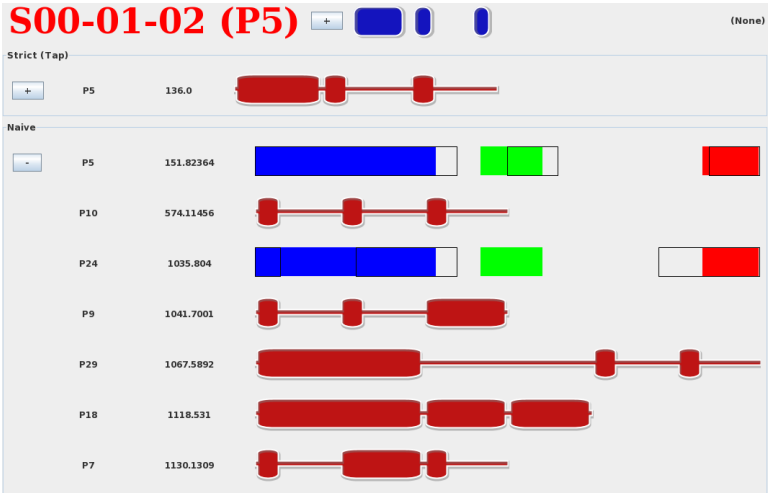


FIGURE 5.10 – *AnEwe, l'outil de test et d'aide à la conception de reconnaisseurs. Il permet de faire reconnaître une séquence de tapes effectuée par un utilisateur par n'importe quel reconnaisseur par rapport à un vocabulaire de motifs rythmiques. Ici, deux des motifs résultats proposés par un reconnaisseur (P5 et P24) sont affichés sous forme de distribution.*

l'exécution des motifs rythmiques par les sujets ne ressortent pas de cette analyse. De plus, les buts de cette première expérience étaient multiples :

- Tester si les participants pouvaient reproduire des motifs rythmique ;
- Développer et valider des reconnaisseurs pour l'interaction rythmique ;
- Commencer à former des intuitions sur les difficultés que pouvait représenter certains motifs ou certaines combinaisons d'évènements/silences.

Si le premier de ces buts est rempli par l'analyse statistique produite dans la section 5.5, les deux derniers ont été réalisés en grande partie à l'aide d'un outil conçu et amélioré tout au long de cette première étude de l'interaction rythmique : AnEwe.

Cet outil était au départ destiné à la conception des reconnaisseurs (Figure 5.10). Il permet de charger des séquences de tapes issues des expériences utilisateurs ou de pilotes réalisés lors de la phase de conception puis de faire classifier ces séquences par n'importe quel reconnaisseur. Il permet donc lors de la conception des reconnaisseurs de rapidement visualiser les possibles aberrations de reconnaissance et d'analyser visuellement les raisons d'échec de reconnaissance.

La sélection des séquences à analyser se fait par le biais d'un système de filtres (voir la Figure 5.11.(a)). Ce système est flexible et il est possible d'ajouter des filtres à l'application. De plus les filtres peuvent être chaînés pour affiner une sélection. Par exemple, ils est possible de sélectionner toutes les séquences dont la

cible était un motif rythmique à 3 évènements, dont la longueur est de 4 pulsations et qui ont été reconnus avec succès.

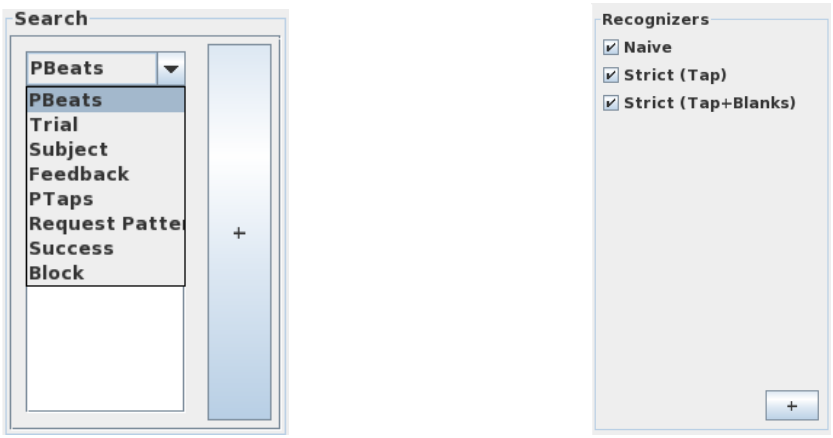
La sélection du (ou des) reconnaisseurs à utiliser pour tenter de trouver le motif du vocabulaire correspond aux séquences de tapes se fait dans un autre panneau (voir la Figure 5.11.(b)). Il est cependant possible d'inclure un autre reconnaisseur simplement en sélectionnant le fichier de code-objet (.class) d'une classe Java appropriée.

Lorsque la sélection des séquences ou des reconnaisseurs à utiliser sont modifiés, l'application met à jour le panneau de résultats. L'application montre alors le motif reconnu par chaque reconnaisseur avec son score de reconnaissance associé. La Figure 5.10 montre un exemple de résultat fourni par l'outil. AnEwe reprend la représentation graphique des motifs telle qu'utilisée comme stimulus dans la première expérience et la représentation graphique des séquences de tapes utilisée comme feedback visuel. En cliquant sur le motif, il est possible de faire apparaître la mise à l'échelle de la séquence par rapport au motif. Sur l'exemple présenté en figure 5.10, deux des résultats issus d'un reconnaisseur ont été transformés en distribution.

Une distribution est, ici, une visualisation simple permettant de facilement apprécier l'adéquation entre une séquence effectuée sur un dispositif d'entrée par un utilisateur pour un motif quelconque. D'abord, le motif est transformé en une séquence de référence dont les durées des événements sont calculées pour un tempo donné (ici 120 pulsations par minutes). Puis, les durées de la séquence sont multipliées par un coefficient de façon à ce que les durées totales de la séquence d'entrée et celle de référence soient les mêmes. Enfin la représentation graphique est obtenue en affichant chaque événement de la séquence d'entrée par un rectangle plein d'une couleur différente pour chaque événement. Finalement les événements de la séquence de référence sont représentés par des bordures de rectangle noir.

Cet outil a permis une exploration plus poussée des enregistrements de la première expérience, particulièrement difficiles à interpréter. Si la séquence sélectionnée est issue d'un enregistrement de l'expérience, alors cliquer sur le bouton "+" permet de faire apparaître le résultat.

Une autre fonctionnalité a été ajoutée à AnEwe pour tenter de comprendre les raisons pour lesquelles certains motifs étaient plus aisés à reproduire que d'autres. Lorsque l'utilisateur sélectionne des enregistrements issus de l'expérience, l'outil groupe les différents essais par rapport au motif dont le protocole expérimental demandait la reproduction à l'utilisateur. L'outil agrège alors les distributions des séquences afin de produire une sorte de *heatmap* de la reproduction de ce motif. La Figure 5.13 présente un exemple de ces distributions agrégées. L'intérêt de cette visualisation est de pouvoir visuellement et rapidement repérer des tendances qu'il est ensuite possible d'analyser à l'aide de statistiques classiques.



(a) (b)

FIGURE 5.11 – *L'outil de filtre pour sélectionner les séquences de tapes à visualiser. Il est possible chaîner les filtres pour raffiner la sélection.*

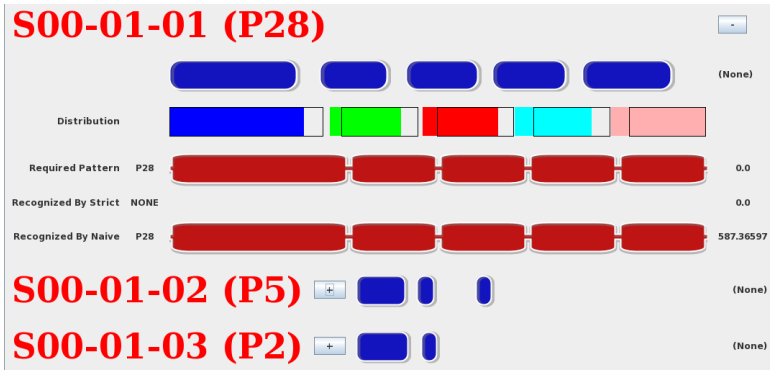


FIGURE 5.12 – *AnEwe permet d'explorer aisément les résultats de la première expérience.*

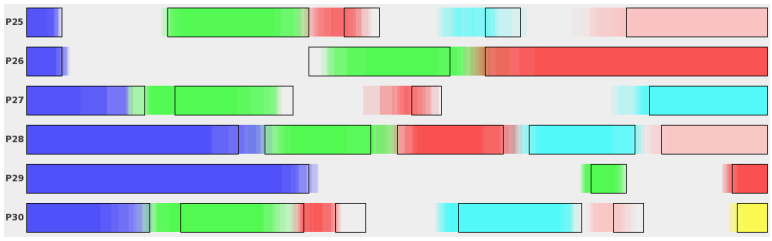


FIGURE 5.13 – *Distributions agrégées issues des enregistrements du sujet S0 de la première expérience utilisateur. Il est possible de voir informellement que ce sujet, bien que très performant, à du mal à positionner des impulsions à l'intérieur d'un motif (P25, P27 et P30).*

Au final, AnEwe a été utilisé lors de la phase d'analyse de l'expérience 1. Sur la Figure 5.13 par exemple, on peut observer le manque de précision dans la reproduction des silences, surtout dans le cas d'impulsions entourées de silences. Puis il a permis de faire fonctionner le classificateur décrit dans la section suivante sur les données de l'expérience 1 pour en valider la conception.

5.7 Un Classifieur de Motifs

Le rôle du reconnaisseur structurel dans la première expérience (présentée dans la section 5.5.1) était d'analyser à quel point les utilisateurs peuvent être précis dans la reproduction d'un motif rythmique. Il était délibérément strict, ne permettant que de petites variations dans le tempo global de la séquence d'entrée. De plus, il ne tire pas avantage de l'information que le motif recherché fait partie d'un vocabulaire restreint choisi à l'avance. Nous avons conçu un autre reconnaisseur plus adapté à l'usage dans un système réel. L'algorithme de ce reconnaisseur classe les motifs du vocabulaire par similitude avec la séquence d'entrée, c'est pourquoi nous l'appelons *classifieur*.

Pour reconnaître une séquence d'entrée, ce *classifieur de motifs* compte tout d'abord le nombre de tapes dans la séquence et extrait du vocabulaire le sous-groupe des motifs possédant le même nombre d'événements. Il calcule ensuite un score de similitude entre la séquence d'entrée et chacun de ces motifs. Pour calculer ce score, il détermine la durée des pulsations en considérant la durée de la séquence d'entrée et la longueur du motif actuellement étudié. En utilisant cette valeur, il convertit le motif en une séquence. Le score de similitude est alors calculé en faisant la somme des différences entre les durées des tapes et des pauses de la séquence d'entrée et de la séquence générée à partir du motif. Une durée d'un quart de pulsation est utilisée pour les impulsions et le relâchement. Finalement, le score est pondéré par le ratio entre la durée d'une pulsation inférée par l'algorithme et la durée d'une pulsation standard à 120 pulsations par minutes (500 *ms*). L'algorithme détaillé, est donné Figure 5.14.

Ce classifieur de motifs est moins strict que le reconnaisseur structurel car il reconnaît toujours un motif pour une séquence d'entrée pourvu qu'au moins un motif du vocabulaire possède le même nombre d'événements que la séquence de tapes. C'est pourquoi, en pratique, un seuil doit être utilisé pour éliminer les scores aberrants. De plus, la normalisation utilisée dans l'algorithme de ce classifieur le rend vulnérable aux motifs homothétiques. D'où l'introduction du score pondéré par la durée de la pulsation standard.

À l'aide de l'outil d'analyse présenté en section 5.6, nous avons testé ce classifieur avec les données et le vocabulaire de l'expérience 1. Le taux de reconnais-

```
events ← countNumberOfEvents(tapSeq)
matchPats ← patternsHavingNTaps(voc, events)
seqDurs ← durations(tapSeq)
results ← Φ

foreach (pat : matchPats) {
    beat ← lengthOfBeat(tapSeq, pat)
    ratio ← 500 / beat
    if (ratio < 1) ratio = 1 + ratio

    patDurs ← durations(pat, beat)

    diff ← 0
    foreach (sDur, pDur : seqDurs, patDurs) {
        diff ← diff + (sDur − pDur)
    }

    score ← diff ∗ ratio

    addResult(results, pat, score)
}

sort(results)
```

FIGURE 5.14 – *Algorithme du classifieur de motifs.*

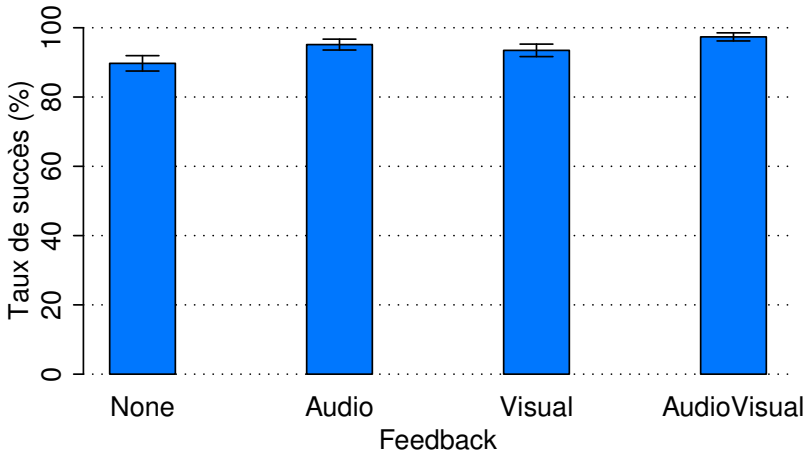


FIGURE 5.15 – *Taux de succès pour chaque* FEEDBACK *avec le classifieur de motifs.*

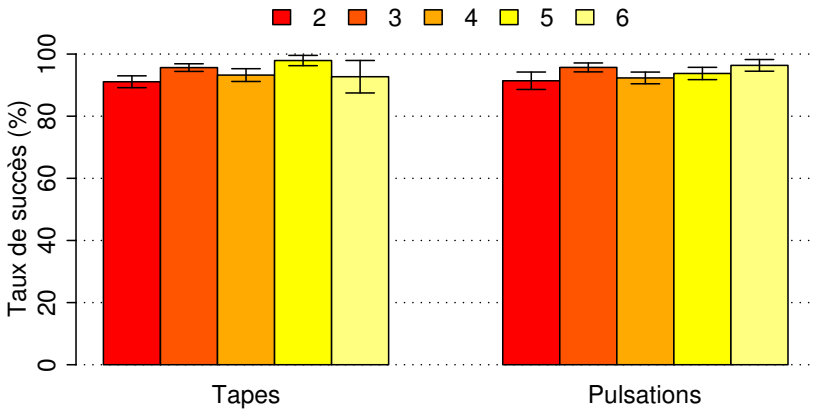


FIGURE 5.16 – *Taux de succès en fonction du nombre de tapes et de la longueur en pulsations du motif avec le classifieur de motifs.*

sance atteint alors 93.9%, un score adéquat pour l'utilisation de l'interaction rythmique dans un contexte applicatif. Comme pour le reconnaisseur structurel, une ANOVA révèle un effet significatif de FEEDBACK sur le taux de succès ($F_{3,33} = 7.2$, $p = 0.0007$, voir la Figure 5.15). Mais dans ce cas le test post-hoc montre une différence significative uniquement entre $None$ et $AudioVisual$.

Par contre, la figure 5.16 montre qu'au contraire du reconnaisseur structurel, le taux de réussite avec notre classifieur ne décroît pas avec la "complexité" du motif : il n'y a pas d'effet significatif du nombre d'événements ou de la longueur du motif sur le taux de succès. A l'inverse, nous observons que le taux de succès est dépendant de la similitude entre certains motifs. Un motif complexe peut donc être aisément reconnu par ce classifieur, pourvu qu'il soit assez différent des autres motifs du vocabulaire possédant le même nombre d'événements. Par exemple, le motif P30 est le seul du vocabulaire à comprendre 6 événements, ce qui fait que le classifieur n'échoue à reconnaître ce motif que si le participant effectue le mauvais nombre de tapes. Au contraire ; P17 est intuitivement plus "complexe" que le "simple" motif P20 mais le premier a un taux de reconnaissance de 100% alors que le second n'obtient que 82%. En fait, le classifieur confond parfois les motifs P20 et P11. Toutefois, une étude a posteriori à l'aide d'un t-test avec une correction de Holm ne révèle aucune différence significative entre les motifs pour ce qui est du taux de succès.

En résumé, nous avons démontré que ce classifieur était adapté à l'utilisation dans une application réelle. Cependant, le concepteur d'une technique utilisant ce reconnaisseur doit prendre garder à choisir un vocabulaire qui minimise les risques de confusion entre les motifs.

5.8 Expérience 2 : Mémorisation des motifs rythmiques

Lors de la première expérience, nous avons étudié la capacité des utilisateurs à reproduire avec assez de précision les motifs rythmiques que nous avons conçus pour un usage en interaction homme-machine. Le but de cette seconde expérience est de tester si de tels motifs rythmiques peuvent être également mémorisés dans l'optique de les utiliser comme alternatives à d'autres techniques plus traditionnelles comme les raccourcis clavier, pour le déclenchement de commandes.

Pour cela nous avons choisi de comparer les motifs rythmiques ($Rhythm$) avec les raccourcis clavier ($Hotkey$) en effectuant une expérience d'apprentissage et de mémorisation inspirée de celle conçue pour comparer l'interaction gestuelle et les raccourcis clavier [6].

5.8.1 Variables

Les sujets pouvaient déclencher des commandes avec l'une des deux techniques : $Hotkey$ et $Rhythm$. Nous avons également intégré une condition se voulant plus réaliste où les participants pouvaient utiliser indifféremment l'une ou l'autre de ces techniques ($Free$).

Quatorze commandes étaient disponibles dans cette expérience (facteur CMD) : C_1, ..., C_{14}. Chacune de ces commandes C_i consistait en un triplet associant une image I_i, servant de stimulus pour la commande, et les deux moyens de déclencher cette commande : un motif rythmique R_i et un raccourci clavier K_i. Nous avons choisi des images représentant les commandes dans un ensemble d'images d'objets du quotidien (banane, gants de boxe, kiwi, tomate, vélo, ... voir la Figure 5.17).

Pour les motifs rythmiques, nous avons utilisé 14 motifs provenant de la première expérience, de complexité variable, que nous avons associé de manière aléatoire aux commandes (images). Chaque raccourci clavier est la combinaison d'un modificateur choisi aléatoirement parmi `Shift` et `Ctrl` et d'une lettre. Les lettres sont choisies de façon à éviter des rapprochements trop aisés avec l'image servant de stimulus.

5.8.2 La Tâche

La tâche principale de l'expérience était d'activer une commande (C_i), représentée par une image utilisée comme stimulus (I_i), à l'aide de la technique d'activation correspondant à la valeur active de TECH (R_i ou K_i). Cette tâche pouvait

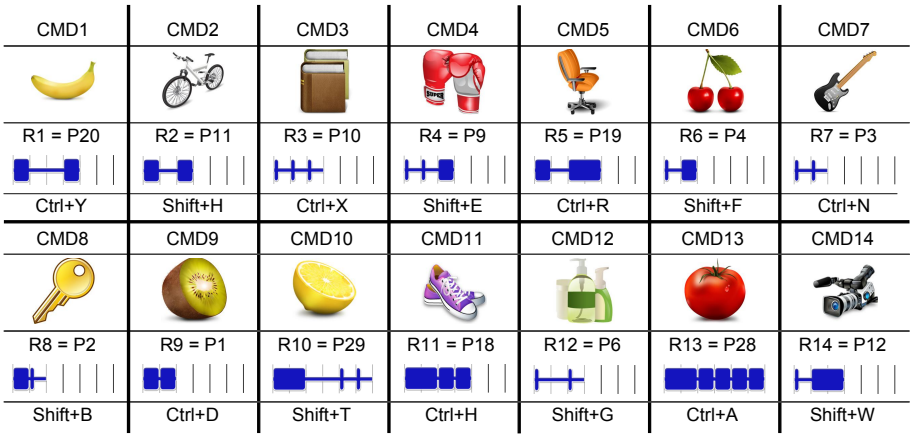


FIGURE 5.17 – *Commandes utilisées dans la seconde expérience. Pxx se rapporte aux motifs utilisé dans la première expérience (Figure 5.5).*

prendre deux formes différentes en fonction de la phase expérimentale : une *forme d'apprentissage* et une *phase de test*.

Pendant la phase d'apprentissage, l'image (I_i) et le déclencheur correspondant (R_i ou K_i) sont affichés simultanément. Pour les motifs rythmiques (R_i), la représentation graphique statique est affichée à côté de l'image (voir la Figure 5.18.a) et la représentation audio du motif est jouée deux fois. Les raccourcis clavier sont présenté à l'utilisateur sous la forme d'une animation de la séquence de touches à effectuer (également jouée deux fois). La représentation textuelle du raccourci clavier est également affichée (voir la Figure 5.18.b). Les techniques d'activation standard telles que les raccourcis clavier et les gestes ont une représentation visuelle "naturelle" qui peut être utilisée dans un aide-mémoire pour aider l'utilisateur à apprendre les commandes. Au contraire, la représentation naturelle des motifs rythmique est basée sur le son. C'est pourquoi nous avons choisi de présenter les deux représentations auditives et visuelles en même temps.

Lors de la phase de test, nous ne présentons aux participants que l'image (I_i) (voir la Figure 5.18.c). Selon la condition fixée par le facteur TECH, les participants doivent effectuer le raccourci clavier (K_i) ou le motif rythmique (R_i) correspondant. Les participants sont fortement encouragés à utiliser l'écran d'aide en appuyant sur la touche ESPACE lorsqu'ils ont oublié le raccourci clavier ou le motif rythmique correspondant. Dans de tels cas, la tâche bascule dans le mode apprentissage tel que décrit précédemment pour cet essai.

Quelle que soit la phase expérimentale active, l'objectif est d'effectuer le motif rythmique ou le raccourci clavier correspondant à la commande demandée. Lorsque les sujets jouent les motifs rythmiques, seul le retour audio est activé suivant les enseignements de la première expérience. Pour les raccourcis clavier,

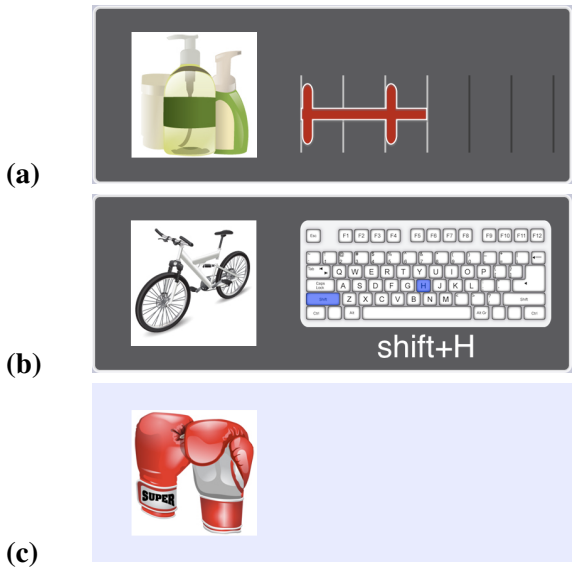


FIGURE 5.18 – ***(a,b)*** *Les Stimulus lors de la phase d'apprentissage (resp.* $Rhythm$ *et* $Hotkey$*).* ***(c)*** *Le stimulus lors de la phase de test.*

aucun feedback graphique ou auditif n'est donné par le système, le feedback kinesthésique fourni par le fait d'appuyer sur les touches du clavier étant jugé suffisant.

La reconnaissance des motif rythmiques est effectuée à l'aide du classificateur de motifs décrit plus haut. Cependant, à la fin de chaque essai de la phase de test, nous demandons également aux participants quel déclencheur ils ont **voulu faire** avant de leur indiquer le résultat de la reconnaissance (voir la Figure 5.19.a pour $Rhythm$ et la Figure 5.19.b pour $Hotkey$). En effet, le but premier de l'expérience est d'évaluer la capacité d'apprentissage des associations, pas la capacité à produire les entrées rythmiques ou textuelles.

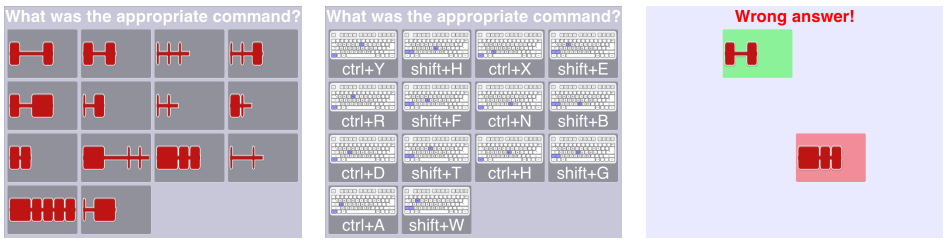


FIGURE 5.19 – *Confirmation dans la condition* $Rhythm$ *(a) et dans la condition* $Hotkey$ *(b). (c) : Feedback donné à l'utilisateur si celui-ci donne la mauvaise réponse (ici, condition* $Rhythm$*).*

5.8.3 Apparatus & Participants

Nous avons utilisé le même matériel que pour la première expérience et nous avons réuni 14 participants, dont 5 femmes. L'âge des sujets s'étendait de 22 ans à 33 ans (moyenne 26 ans et médiane à 26 ans). Cinq de ces sujets avaient déjà pris part à la première expérience.

5.8.4 Procédure et schéma expérimental

L'expérience suit un schéma expérimental intra-sujets avec comme facteurs principaux la technique d'activation (TECH) et les commandes (CMD). L'expérience est découpée en deux sessions ayant lieu sur deux jours consécutifs. Le premier jour, tous les participants doivent se familiariser avec la reproduction des motifs rythmiques en effectuant un entrainement de 5 minutes reprenant la tâche de la première expérience. Nous utilisons (TECH) comme facteur de bloquage tout en le contrebalançant entre les participants. Le deuxième jour, nous ajoutons un nouveau bloc : ($Free$) apparenté aux blocs de test, mais ne disposant pas de la possibilité d'aide. Il appartenait à l'utilisateur de choisir la technique de déclenchement ($Hotkey$ ou $Rhythm$) qu'il préfère.

Chaque bloc TECH est constitué de sous-blocs de 15 essais chacun :

- 2 sous blocs d'entrainement et 2×4 blocs de tests pour le premier jour.
- 4 sous blocs de tests pour la deuxième session.

La phase de test de l'expérience est donc découpée en deux sous-sessions SUBSESSION de 60 essais chacune. Le premier jours pour évaluer la mémorisation immédiate des raccourcis de commandes, et le deuxième jour pour tester la mémoire à moyen terme (voir la Figure 5.20).

Dans le but de simuler une utilisation plus réaliste, dans laquelle des commandes sont plus utilisées que d'autres, nous avons associé à chacune des 14 commandes une fréquence d'apparition suivant une distribution de Zipf en nous inspirant d'expériences antérieures [47, 6]. Lors de la phase d'apprentissage nous utilisons les fréquences suivantes : (6, 6, 3, 3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1). Lors de la phase de test nous utilisions des fréquences doublées : (12, 12, 6, 6, 4, 4, 3, 3, 2, 2, 2, 2, 1, 1). Nous associons les fréquences aux commandes selon 7 permutations qui sont contre-balancées entre les participants, donnant le même nombre de mesures pour toutes les commandes (pour l'ensemble des sujets). L'apparition des commandes est alors aléatoire au sein d'une SUBSESSION.

Chaque participant a mis environ une heure pour effectuer la première session et 30 minutes pour la seconde. À la fin de chaque session, un questionnaire était donné aux sujets afin de collecter des réactions subjectives de leurs part.

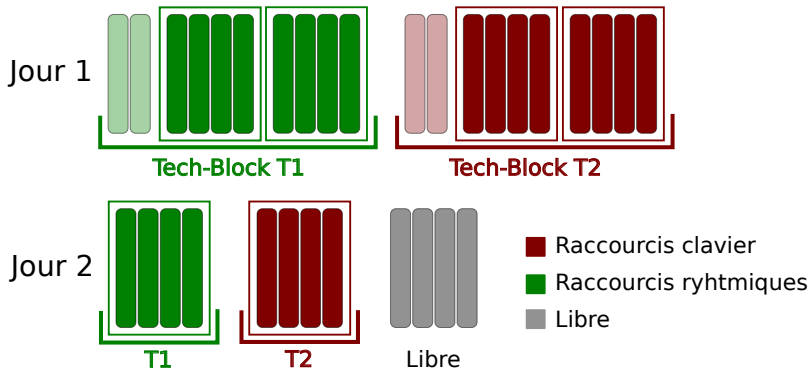


FIGURE 5.20 – *Exemple de session pour un participant. Les blocs de couleur moins saturée sont des blocs d'entrainement. Les blocs entourés représentent les sous-sessions.*

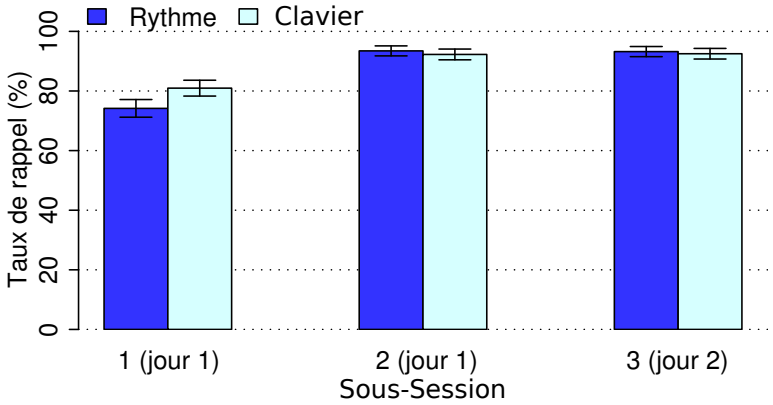


FIGURE 5.21 – *Taux de rappel pour les deux techniques pour chaque sous-session.*

5.8.5 Résultats Quantitatifs

Nous mesurons principalement deux variables lors de l'expérience :

- Le *taux de rappel*, défini comme le pourcentage de réponses correctes lors de la phase de test sans avoir utilisé l'aide ;
- Le *taux d'aide* désignant le pourcentage de fois que le participant à utilisé l'aide pendant la phase de test.

Nous analysons les résultats par rapport à TECH et aux trois sous-sessions de l'expérience : TECH $\times$ SUBSESSION $\times$ Rand(PARTICIPANT).

L'analyse révèle un effet significatif de SUBSESSION sur le taux de rappel ($F_{2,26} = 103$, $p < 0.0001$). Une analyse post-hoc utilisant un t-test avec correction

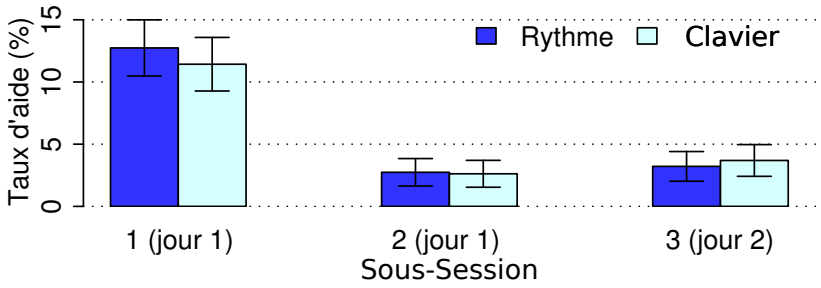


FIGURE 5.22 – *Taux d'utilisation de l'aide pour les deux techniques en fonction de la sous-session.*

de Bonferroni indique que les participants ont eu plus de problèmes à mémoriser les associations lors de la première sous-session que lors des deux suivantes (voir la Figure 5.21).

L'analyse ne montre pas d'effet significatif de TECH sur ce taux de rappel ($F_{1,13} = 0.61$, $p = 0.4474$), cependant une ANOVA montre une interaction significative pour TECH $\times$ SUBSESSION ($F_{2,26} = 5.36$, $p = 0.0113$). Une analyse post-hoc utilisant un t-test avec correction de Bonferroni montre une différence significative entre $Rhythm$ et $Hotkey$ pour la première sous-session (74% et 81% respectivement), mais le taux de rappel pour les deux techniques est très proche pour les deux sous-sessions restantes, avoisinant 93% de mémorisation (voir la Figure 5.21).

Concernant l'utilisation de l'aide, l'ANOVA montre un effet significatif de SUBSESSION ($F_{2,26} = 17.3$, $p < 0.0001$), aucun effet de TECH ($F_{1,13} = 0.04$, $p = 0.8532$) et enfin aucune interaction significative entre TECH $\times$ SUBSESSION ($F_{2,26} = 0.62$, $p = 0.545$). Nous n'avons trouvé qu'une seule différence significative entre les sous-sessions : l'aide a été plus utilisée lors de la première sous-session que dans les deux restantes (voir la Figure 5.22).

L'analyse des résultats pour chaque commande donne des résultats similaires. En ce qui concerne $Rhythm$ une ANOVA pour le modèle CMD $\times$ SUBSESSION $\times$ Rand(PARTICIPANT) pour le taux de rappel montre un effet significatif de CMD ($F_{13,169} = 1.17$, $p = 0.025$). Un t-test post-hoc avec correction de Holm montre seulement une différence significative entre R_3 et R_{13} (taux de rappel d'environ 97%) et R_{10}, R_{11} et R_{14} ($\sim$80%).

Pour tester notre classifieur, nous avons comparé les motifs reconnus par le classifieur avec la réponse donnée par les participants en utilisant le modèle TECH $\times$ SUBSESSION $\times$ Rand(PARTICIPANT). L'ANOVA montre un effet significatif de TECH ($F_{1,13} = 5.34$, $p = 0.038$), $Rhythm$ ayant un taux de succès inférieur à celui de $Hotkey$: 85.2% vs. 91.8%. Le taux de succès pour $Hotkey$ est plus bas que l'on aurait pu l'imaginer : on pouvait s'attendre à un taux de succès proche de

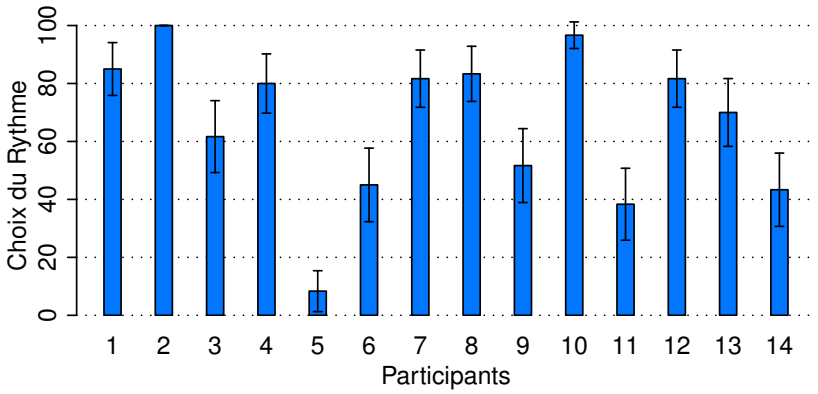


FIGURE 5.23 – *Pourcentage des essais effectués à l'aide de $Rhythm$ lors de la phase de test libre.*

100%. Ceci est probablement dû au fait que les participants ont changé d'avis au moment où on leur montrait l'écran de réponse. Pour $Rhythm$ le taux de succès est aussi inférieur à celui attendu. Mais en fait, le taux de succès pour $Rhythm$ relativement à $Hotkey$ est de 92.8%, proche du taux obtenu avec les données de la première expérience (94%).

Les résultats des deux techniques $Rhythm$ et $Hotkey$ sont similaires, indiquant que l'association de motifs rythmiques à des commandes à activer pourrait être dans une utilisation réelle aussi performante que l'association avec une technique classique telle que celle des raccourcis clavier. Ce résultat est remarquable état donné l'utilisation très répandue des raccourcis clavier.

5.8.6 Résultats Qualitatifs

La Figure 5.23 montre le pourcentage de fois où les participants ont choisi d'utiliser les motifs rythmiques lors de la phase libre de la second session. Sept participants ont utilisé les motifs rythmiques plus de 80% du essais et un participant a uniquement a utilisé cette technique dans moins de 20% des essais libres.

Les réponses des participants aux questionnaires étaient globalement positives, confirmant les résultats précédents. Parmi les 14 participants, 9 ont préféré utiliser les motifs rythmiques, 3 ont préféré l'utilisation des raccourcis clavier et deux n'avaient aucune préférence. La raison invoquée pour préférer les motifs rythmiques était le côté "ludique" de la reproduction des rythmes, mais aussi le fait que les motifs rythmiques pouvaient être exécutés sur place sur le pavé tactile sans avoir à rechercher les touches (modifieur et lettre) sur le clavier. À l'inverse, certains sujets nous ont rapporté que les raccourcis clavier étaient plus rapides à exécuter et qu'ils les préféraient donc lorsque le motif rythmique était trop long.

En ce qui concerne la mémorisation, un point intéressant soulevé par les participants était l'utilisation de mnémoniques en correspondance directe avec le rythme afin d'aider à mémoriser le motif. Par exemple, un sujet à associé la commande "gants de boxe" du motif 29 (voir Figure 5.5) à l'onomatopée "pif pam poum" qui rappelle la structure "court court long" du motif. Un autre participant a expliqué qu'il tentait de lier la structure du motif à la prononciation du mot représenté sur l'image (par exemple "toma-to-to-to-to" pour la commande 13 et le motif P28). Les sujets ont également utilisé la représentation graphique des motifs afin de les mémoriser, comme un participant rapportant que "le motif correspondant à la cerise à une représentation ressemblant à une cerise". Ces commentaires indiquent que les participants sont capables d'élaborer des techniques de mémorisation efficaces des motifs rythmiques, non seulement basé sur les rythmes, mais également sur notre représentation graphique. Ceci renforce l'intuition que nous avons que notre représentation graphique pourrait être utilisée dans des aide-mémoire comme pour les gestes ou les raccourcis clavier.

5.9 Conclusion

L'interaction rythmique telle que décrite dans ce chapitre présente une opportunité pour généraliser les techniques existantes basées sur le temps telles que le clic long ou le double clic. Elle est également et avant tout une alternative aux méthodes d'entrée plus traditionnelles, reposant sur le temps plutôt que l'espace. Dans cette optique, un *langage simple* a été présenté, en combinaison avec deux *reconnaisseurs universels* pour les motifs rythmiques créés à partir de ce langage. L'un des avantages de ces reconnaisseurs est qu'ils ne requièrent pas d'apprentissage.

La première expérience utilisateur a permis d'évaluer la capacité des utilisateurs à reproduire précisément des motifs rythmiques avec différents feedbacks. Elle montre que même si des motifs complexes sont difficiles à reproduire, un feedback audio et/ou visuel permet d'accroître les performances. De plus, même avec l'utilisation d'un vocabulaire conséquent de 30 motifs, le taux de reconnaissance peut atteindre 93.9% avec un reconnaisseur prenant en compte les difficultés de reproduction. Devant les résultats encourageants de la première expérience, une seconde expérience a été réalisée dans le but d'étudier les possibilités de mémorisation des motifs rythmiques en association avec des commandes. Ce type d'utilisation de l'interaction rythmique pourrait être qualifiée de *raccourci rythmique* par analogie avec les raccourcis clavier. Les résultats de cette seconde étude montrent que les raccourcis rythmiques ont des performances de mémorisation similaires à celles des raccourcis clavier.

Ces deux expériences contrôlées sont encourageantes et permettent d'envisager l'utilisation des motifs rythmiques comme une méthode d'entrée. Il est évident

qu'une telle méthode, bien qu'applicable dans certaine condition pour le déclenchement de commandes dans un environnement de bureau ne peut et ne doit pas remplacer l'utilisation de méthodes d'entrées plus adaptées dans ce type d'environnement. Par contre, cette approche peut permettre une interaction plus aisée dans d'autres contextes : le contrôle aveugle d'un appareil mobile tel qu'un téléphone cellulaire ou un baladeur, par exemple dans la poche, ou le contrôle distant d'un environnement interactif, comme un mur d'images, en jouant les motifs sur des capteurs portables.

Enfin, il est envisageable de généraliser cette approche en conjuguant la dimension rythmique avec d'autres dimensions ou techniques d'interaction. Des actions plus complexes que le simple fait de taper du doigt en respectant une structure rythmique sont possibles comme effectuer des gestes ou presser des touches du clavier, la dimension rythmique permettant alors d'ajouter de l'information. Dans une moindre mesure, l'interaction permettant l'accès aux historiques présenté dans la section 2.2 est un exemple de cette proposition. Les interactions proposées sont les mêmes pour les opérations standard de collage du presse papier et de la sélection primaire du système X Window et pour ces mêmes opérations augmentées des historiques correspondants. Les deux familles de commande ne sont différentes qu'à une composante temporelle près, un des motifs rythmiques les plus simples, permettant au système de fenêtrage de détecter si l'utilisateur veut communiquer avec lui (historiques) ou avec la fenêtre active (interactions standard).

CHAPITRE 6

Conclusion et Perspectives

L'objectif de cette thèse est de reconsidérer l'interaction avec les systèmes interactifs en général et avec les gestionnaires de fenêtres en particulier en mettant l'accent sur la stabilité de l'espace d'affichage et la conservation de l'attention de l'utilisateur, préconisant une interaction localisée. Bien que ces concepts soient connus des psychologues et utilisés en interaction homme-machine, ils ont rarement été appliqués dans le cadre des gestionnaires de fenêtres.

La démarche suivie au cours de cette thèse a commencé par la conception de techniques d'interaction simples, implémentées dans un environnement réel et pouvant être utilisées au quotidien. Cette phase de création a été propice au questionnement sur la façon dont les utilisateurs perçoivent leur environnement de travail. La suite des travaux présentés dans cette thèse étudie des points particuliers du contexte spatial qui étaient importants dans les techniques proposées dans la phase initiale (chapitre 2).

Nous rappelons ci-après les principales contributions apportées par cette thèse et nous donnons des directions possibles pour poursuivre ces travaux.

6.1 Stabilité Spatiale

6.1.1 Le problème

L'être humain a une conception innée de l'espace. Il est capable de retrouver des objets dans un environnement complexe. Il est également doué pour se représenter l'espace dans lequel il évolue en étant par exemple capable de trouver son chemin dans une ville inconnue à l'aide d'une carte. Cependant, lors du passage dans l'univers immatériel qu'est l'environnement de bureau informatique cette capacité est parfois ignorée ou sous-employée. Les opérations concernant la gestion des fenêtres proposées dans ce type d'environnement sont minimalistes et requièrent des efforts de la part de l'utilisateur. Même si l'utilisateur porte un soin

tout particulier à organiser ses éléments de travail, certains outils offerts par le système les ignore en proposant des techniques tenant peu compte de cette capacité à se remémorer aisément l'emplacement des objets.

Toutefois, l'espace proposé par un environnement de bureau est différent de celui dans lequel évolue l'être humain. L'espace réel possède trois dimensions. Une photo ou une image en possède deux. Contrairement à la surface d'affichage utilisée pour afficher l'environnement de l'utilisateur, la majorité des gestionnaires de fenêtres actuels utilisent non pas un espace plan en deux dimensions, mais plutôt un espace qu'il est possible de qualifier de 2D 1/2. En effet, en permettant la superposition des fenêtres, ils introduisent une troisième dimension, différente de la 3D puisqu'il n'existe pas à proprement parler de volume. Cependant, l'effet de cette dimension, appelée dans cette thèse profondeur par similitude avec la profondeur en 3 dimensions, n'a été que peu étudiée en Interaction Homme-Machine. L'utilisateur perçoit-il cette profondeur avec assez de vivacité pour pouvoir la gérer efficacement, voire la tourner à son avantage et s'en servir dans la gestion de son espace de travail ?

Le mode d'interaction utilisé pour interagir avec cet univers est également différent de celui du monde réel. L'acquisition d'un objet à l'aide d'un dispositif de pointage est encore à l'heure actuelle l'une des principales sources d'interaction entre l'utilisateur et son environnement de bureau. Cliquer sur un icone, sélectionner le point d'insertion dans un texte, choisir un élément dans un menu sont autant d'actions faisant appel à la capacité de pointer. Comprendre les mécanismes de cette action de pointage est nécessaire à la compréhension de l'espace dans lequel évolue l'utilisateur. Le cas du pointage sur des cibles statiques a été abondamment étudié en interaction homme-machine. Mais le cas statique n'est pas le seul, car il arrive bien souvent que des cibles apparaissent soudainement (bouton d'une fenêtre de dialogue) ou se déplacent à l'écran via des animations (zone de dépôt pour un icone). Il est alors naturel d'étudier le pointage dans ce type de condition. Quelle différence peut-il exister entre l'acquisition de cibles statiques et l'acquisition de cibles en mouvement ou apparaissant subitement ? S'il existe des différences, quelles sont les caractéristiques à prendre en considération lors de la conception d'interfaces graphiques ? Ces questions sont importantes pour mieux comprendre l'impact de la mémoire spatiale, des animations et de l'apparition d'objets à l'écran sur le pointage, qui constitue la brique des base de tout dialogue de plus haut niveau.

6.1.2 Les Contributions

Les interactions *DeskPop* et *StackLeafing* présentées dans la première partie de cette thèse ont été conçues avec comme but principal de déplacer au minimum les objets affichés dans l'environnement de travail de l'utilisateur. À cette fin, ces

interactions exploitent la profondeur des objets. *DeskPop* est une technique permettant d'afficher et d'interagir avec le bureau tout en conservant les fenêtres visibles. Elle est donc une alternative à des techniques comme *Show Desktop* qui cache les fenêtres ou l'un des modes d'*Exposé* de Mac OS X qui les déplaces et les transforme. *StackLeafing* est une technique offrant la possibilité de parcourir les fenêtres (et le bureau) pour trouver celle qui intéresse l'utilisateur. Son originalité tient à son utilisation du non-chevauchement entre fenêtres pour constituer des groupes de fenêtres appelés couches. Au lieu de parcourir les fenêtres une par une, l'utilisateur peut alors les parcourir couche par couche. Les fenêtres d'une couche ne se recouvrant pas, l'utilisateur garde ainsi une visibilité complète des fenêtres.

Au-delà des effets sur le confort d'utilisateur, l'animation des objets à l'écran ou leur masquage même pour de courtes durées impactent-ils les performances de l'utilisateur, même pour une tâche aussi simple que le pointage à l'aide d'un dispositif ? Il s'avère que l'animation ou l'apparition soudaine d'une cible reste très proche au niveau des performances de pointage du cas de base (cible statique toujours affichée) si cette animation ou le délai d'apparition après le début du pointage est très court (en dessous de 200 millisecondes). Cependant, le cas non statique conduit à plus d'erreurs et, dès que le délai dépasse ces 200 millisecondes, des différences entre le pointage sur cibles statique et le pointage sur cibles animée ou surgissante commencent à apparaître. Ce résultat montre que des techniques telles que *StackLeafing* et *DeskPop* qui conservent les cibles potentielles statiques et visibles possèdent un avantage.

Une particularité de *DeskPop* et de *StackLeafing* est d'utiliser et de modifier la profondeur des fenêtres. Cette profondeur a été peu étudiée dans le cas de l'espace en 2D 1/2 proposé par les gestionnaires de fenêtres. Toutefois, l'importance de son utilisation dans les techniques alternatives proposées dans ce manuscrit amène à se poser la question de savoir s'il est possible d'améliorer la perception de la profondeur des fenêtres. À cette fin, trois Indices Visuels de Profondeurs – luminosité, flou et ombre – ont été testés lors d'une expérience utilisateur. Il ressort de cette étude que la luminosité est un candidat intéressant pour permettre de mieux appréhender la profondeur en gestion des fenêtres. Les deux autres indices envisagés lors de cette expérience n'ont pas réellement prouvé leur efficacité.

Parallèlement à ces résultats fortement liés aux techniques d'interaction *StackLeafing* et *DeskPop*, nos études avaient également pour but une meilleure compréhension du contexte spatial en général. Elles ont permis de découvrir des faits intéressants :

– Les utilisateurs adaptent leur mouvement de pointage au comportement de la cible : lorsque l'utilisateur essaie d'acquérir une cible animée d'une translation, son mouvement est plus lent (au pic de vitesse) et plus saccadé que s'il essayait d'acquérir une cible surgissante, comme s'il "suivait" la cible.

Au final, les mouvements de pointage pour ces deux comportements de cible sont assez différents.
– Nous avons montré que l'on pouvait étendre la loi de Fitts avec un modèle possédant une interprétation intuitive pour modéliser l'acquisition de cibles animées et surgissantes en rajoutant un terme fonction du délai d'animation ou d'apparition et de la distance de pointage.
– Même lors de l'utilisation d'un système par couche, la complexité d'une scène dans un gestionnaire de fenêtres reste le nombre global de fenêtres et non celui de couches.

6.1.3 Les Perspectives

Le contexte spatial, même réduit à la gestion de fenêtres, est un sujet vaste et qui mérite d'être exploré plus avant. Nos travaux, bien que participant à une meilleure compréhension du contexte spatial, ne sont que préliminaires. Les études présentées dans cette thèse ont été volontairement réalisées avec des tâches abstraites, pour plus de généralité. Si une telle approche nous a éloignés d'une interaction réaliste, cela nous a permis d'étudier les facteurs fondamentaux et de fait nous a donné des pistes à prendre en compte pour l'étude de cas d'utilisation plus concrets.

Notre étude de la profondeur a été réalisée sur des scènes statiques. Or, dans l'utilisation d'un gestionnaire de fenêtres, les fenêtres changent de profondeur. Une étude de l'aspect dynamique des indices visuels de profondeur nous semble être une prolongation naturelle de nos travaux. Les indices utilisant la luminosité et le flou sont continus et peuvent donc être animés (e.g., variation progressive de luminosité). Cela suffit-il à aider l'utilisateur à mieux percevoir les changements de profondeur ? Pour sa part, l'indice visuel de profondeur fondé sur les ombres est constant quelque soit la profondeur de la fenêtre, du moins pour la version utilisée dans notre étude et les ombres portées généralement rencontrées. Il est cependant envisageable de modifier également l'ombre de la fenêtre selon sa profondeur (voir True Shadows de Dragicevic [1]). Il est également possible d'imaginer des animations spatiales lors du changement de profondeur. De telles animations pourraient, par exemple, utiliser le pliage des fenêtres à la manière de Fold'n'Drop, une fenêtre passant dernière une autre se pliant puis se dépliant pour laisser passer l'autre au premier plan. Quels sont les avantages de telles animations (IVP et spatiales) ? En particulier, l'animation de la luminosité peut-elle améliorer à la fois la perception statique de la profondeur et celle du changement de profondeur ? Enfin, quel serait l'impact d'animations spatiales lors d'une modification de la profondeur d'une fenêtre ?

1. http://www.lri.fr/~dragice/trueshadows/

De même, l'étude du pointage sur les cibles animées ou surgissantes a volontairement été restreinte à un pointage en une dimension sans distracteurs. L'interaction avec un ordinateur se fait toutefois en deux dimensions et dans un environnement plus complexe. Maintenant que nous disposons d'une bonne compréhension des effets des facteurs fondamentaux, il serait intéressant d'étudier les conséquences d'un passage à une tâche en deux dimensions et surtout plus réaliste, avec des fenêtres et des icones. On pourrait, par exemple, étudier la phase de dépôt d'un icone dans une fenêtre animée, ou encore étudier le cas des menus déroulants afin de voir si le comportement concorde avec celui que nous avons observé dans nos travaux pour les cibles surgissantes.

6.2 Localité de l'Interaction

6.2.1 Le Problème

La littérature en interaction homme-machine abonde de techniques d'interaction innovantes. Malheureusement, ces interactions – pour des raisons évidentes de simplicité et de contraintes liées à la recherche – sont souvent réalisées sous la forme de prototypes autonomes. La question du déclenchement de ces interactions, nécessaire à son intégration dans un système réel est donc parfois laissée en suspend. Inversement, dans les systèmes industriels où l'intégration des fonctionnalités est un impératif, des solutions de contournement ont été utilisées. Souvent ces solutions reposent sur l'utilisation de raccourcis clavier avec deux modificateurs pour réduire la possibilité d'interférence avec les applications. Ce type de raccourci est à la fois complexe à apprendre et à réaliser. Ils ont également le désavantage d'obliger l'utilisateur à changer de modalité d'entrée s'ils utilisaient la souris. Un autre procédé souvent utilisé est l'ajout d'un bouton permettant d'accéder à certaines fonctionnalités additionnelles. Ce type de boutons est souvent placé sur un bord de l'écran ou dans un coin (system tray sous Windows et X Window, barre de menus sous Mac). Interagir avec ces boutons demande à l'utilisateur de déporter son attention.

Au final, nous constatons que la diversité des actions à exécuter lors de l'utilisation d'un système informatique et la multitude de fonctionnalités offertes par les logiciels exigent une grande puissance d'expression du vocabulaire d'entrée de l'utilisateur. Traditionnellement, les canaux d'entrée d'un système informatique sont composés d'un dispositif de pointage (souris, touchpad, touch screen...) et d'une entrée discrète possédant un nombre fini d'actions possibles (le clavier). Cependant, les interactions traditionnelles avec ces dispositifs ne suffisent souvent plus à couvrir les besoins des utilisateurs. Plus exactement, les applications s'accaparent une large partie des interactions standards. Le choix du déclencheur pour une nouvelle fonctionnalité devant être intégrée dans un système de fenêtrage est

alors difficile puisque les fonctionnalités du système ne doivent pas interférer avec les applications.

6.2.2 Les Contributions

Dans le chapitre 2 nous avons utilisé le temps comme dimension d'interaction afin d'intégrer certaines des fonctionnalités que nous proposions. L'utilisation du temps dans une telle situation est avantageuse, car, premièrement et bien qu'il soit utilisé en quelques occasions (e.g. *spring-loaded widgets*), le temps est assez peu utilisé par les applications, deuxièmement, il constitue une dimension orthogonale à l'espace et enfin, il peut être utilisé sur tous les canaux d'entrée (souris, clavier) de manière consistante puisque l'expression du temps ne nécessite en définitive qu'un degré de liberté. Dans *PowerTools*, ces caractéristiques du temps ont été mises à profit pour créer des déclencheurs pour nos interactions en utilisant une règle simple : les interactions normales sont à destination des applications, les interactions temporisées (ou longues) sont adressées au système de fenêtrage. Tout d'abord, l'utilisation agnostique du dispositif nous à permis de prolonger de façon consistante à travers le clavier et la souris les interactions classiques de copier-coller et de sélection primaire de X Window par exemple. L'utilisation du temps, en lieu et place de l'espace, nous a permis de garder l'interaction localisée autour du point d'intérêt, évitant les changements potentiels de contexte spatial.

Par la suite, le travail effectué pour intégrer les PowerTools dans Métisse s'étant révélés prometteurs, nous souhaitions étendre et généraliser ces travaux pour une utilisation plus large et moins ad-hoc. Dans la continuité des l'interaction temporisée de PowerTools, l'interaction rythmique présentée dans cette thèse (travail réalisé en collaboration avec Emilien Ghomi) est une nouvelle forme d'interaction privilégiant la dimension temporelle à la dimension spatiale. Les travaux présentés dans ce manuscrit proposent une grammaire volontairement limitée par rapport à la puissance expressive des rythmes musicaux. Le but était en effet de concevoir une méthode d'entrée utilisable par des utilisateurs novices afin de la rendre adaptée à l'Interaction Homme-Machine. Deux évaluations préliminaires viennent valider notre approche de l'interaction rythmique.

La première expérience utilisateur avait deux buts. Tout d'abord, s'assurer que les motifs dérivés de nos contraintes étaient effectivement assez simples pour pouvoir être reproduits par des utilisateurs, même novices. Ensuite, étudier l'impact de différents feedbacks lors de la reproduction de ces motifs. Pour cette expérience, un reconnaisseur "structurel" et strict a été conçu. L'expérience a confirmé l'intuition que l'utilisation d'un feedback lors de la reproduction de motifs rythmiques améliore la qualité de reproduction, avec cependant le résultat surprenant qu'un feedback visuel et audio n'est pas préféré par les utilisateurs. Parallèlement, les résultats de l'expérience montrent que notre reconnaisseur strict détecte beaucoup

d'erreurs faites par les utilisateurs, notamment sur les pauses entre les tapes. Cependant, l'aspect structurel de ce reconnaisseur le rend indépendant du vocabulaire utilisé lors de l'expérience, les motifs joués par les participants etant recherchés dans l'espace complet des motifs rythmiques (799 motifs) plutôt que les 30 motifs utilisés dans l'expérience. Si le reconnaisseur structurel strict était un excellent candidat pour une expérience de reproduction où nous voulions connaitre les capacités d'utilisateurs novices à reproduire effectivement des motifs, les résultats de l'expérience montre clairement que son utilisation dans un système réel engendrerait, en pratique, beaucoup trop d'erreurs de reconnaissance. Nous proposons donc un autre reconnaisseur laissant plus de place aux erreurs de reproduction et dépendant du vocabulaire : un classifieur de motifs. L'analyse des enregistrements à l'aide de ce reconnaisseur démontre que la reproduction de motifs rythmiques peut être réalisée assez fidèlement pour qu'ils soient reconnus par un système interactif.

Pour une utilisation concrète de cette nouvelle méthode d'entrée en tant que déclencheur potentiel de techniques au sein d'un système interactif, la reproductibilité des motifs rythmiques est bien sûr nécessaire, mais elle n'est pas suffisante. Il faut que les utilisateurs soient également capables de se souvenir des motifs et de leur association avec les actions ou commandes qu'ils désirent effectuer. C'est ce que teste la seconde expérience utilisateur dans laquelle nous comparons le déclenchement de commande par un racourci clavier ou par un motif rythmique. Notre classifieur de motifs rythmiques est utilisé dans cette expérience pour réaliser la reconnaissance des séquence de tapes jouées par les participants. Il ressort de cette expérience que l'association de motifs rythmiques et de commandes est aussi aisée à retenir que l'est l'association de raccourcis clavier et de commandes. Les résultats de cette seconde expérience contrôlée font également état d'une bonne réception de cette méthode d'entrée par les sujets qui la considèrent comme ludique et parfois plus facile à mémoriser que les raccourcis clavier.

6.2.3 Les Perspectives

Les travaux portant sur l'interaction rythmique rapportés dans ce manuscrit ne sont qu'une première approche qui révèle beaucoup de pistes qui nécessitent encore d'être explorées. L'une d'entre elles relève de l'annulation d'une interaction rythmique. Dans un scénario où un utilisateur commence un motif rythmique et s'aperçoit soit d'une erreur de reproduction soit qu'il s'est tout simplement trompé de motif, quelles sont ses possibilités d'interrompre son interaction ou de la corriger dynamiquement sans que le système n'essaie de l'interpréter ? Une autre piste concerne les dispositifs d'entrées. Dans l'étude de l'interaction rythmique, nous avons uniquement considéré le pavé tactile d'un ordinateur portable. Il serait intéressant d'étudier la reproduction des motifs rythmiques sur un plus large panel de dispositifs, en particulier mobiles. D'autres type de feedback pourraient alors également être étudiés comme la vibration ou un autre feedback tactile, afin de pouvoir

utiliser les rythmes dans des cas où un feedback visuel ou auditif n'est pas envisageable. Finalement, l'utilisation réelle de cette méthode d'entrée, par exemple son intégration au niveau du système dans un environnement de bureau ou son usage comme méthode d'entrée sur un dispositif mobile, sont des étapes importantes pour la validation plus écologique de l'interaction rythmique.

Bibliographie

[1] J. Accot and S. Zhai. Beyond fitts' law : models for trajectory-based hci tasks. In *Proc. CHI '97*, pages 295–302. ACM, 1997.

[2] J. Accot and S. Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proc. CHI '02*, pages 73–80. ACM, 2002.

[3] J. Accot and S. Zhai. Refining Fitts' law models for bivariate pointing. In *Proc. CHI '03*, pages 193–200. ACM, 2003.

[4] M. Aivar, M. Hayhoe, C. Chizk, and R. Mruczek. Spatial memory and saccadic targeting in a natural task. *Journal of Vision*, 5(3 :3) :177–193, 2005.

[5] C. Appert and M. Beaudouin-Lafon. SwingStates : Adding state machines to Java and the Swing toolkit. *Software : Practice and Experience*, 38(11) :1149 – 1182, 2008.

[6] C. Appert and S. Zhai. Using strokes as command shortcuts : cognitive benefits and toolkit support. In *Proc. CHI '09*, pages 2289–2298. ACM, 2009.

[7] G. J. Badros, A. Borning, and P. J. Stuckey. The cassowary linear arithmetic constraint solving algorithm. *ACM Trans. Comput.-Hum. Interact.*, 8 :267–306, 2001.

[8] G. J. Badros, J. Nichols, and A. Borning. Scwm : An extensible constraint-enabled window manager. In *Proc. USENIX '01 (FREENIX Track)*, pages 225–234. USENIX Association, 2001.

[9] T. Baudel and M. Beaudouin-Lafon. Charade : Remote control of objects using free-hand gestures. *Communications of the ACM*, 36 :28–35, 1993.

[10] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick : Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. INTERACT '03*, pages 57–64. IOS Press, IFIP, 2003.

[11] P. Baudisch and C. Gutwin. Multiblending : displaying overlapping windows simultaneously without the drawbacks of alpha blending. In *Proc. CHI'04*, pages 367–374. ACM, 2004.

[12] M. Beaudouin-Lafon. Novel interaction techniques for overlapping windows. In *Proc. UIST '01*, pages 153–154. ACM, 2001.

[13] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps : Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21 :833–854, October 2002.

[14] B. A. Bell and S. K. Feiner. Dynamic space management for user interfaces. In *Proc. UIST '00*, pages 239–248. ACM, 2000.

[15] M. Bernard and F. Jacquenet. Free space modeling for placing rectangles without overlapping. *J. Univ. Comp. Sci*, 3 :703–720, 1997.

[16] M. S. Bernstein, J. Shrager, and T. Winograd. Taskposé : exploring fluid boundaries in an associative window visualization. In *Proc. UIST '08*, pages 231–234. ACM, 2008.

[17] A. Bezerianos and R. Balakrishnan. The vacuum : facilitating the manipulation of distant objects. In *Proc. CHI '05*, pages 361–370. ACM, 2005.

[18] S. A. Bly and J. K. Rosenberg. A comparison of tiled and overlapping windows. *SIGCHI Bull.*, 17 :101–106, 1986.

[19] O. Bock and R. Eckmiller. Goal-directed arm movements in absence of visual guidance : evidence for amplitude rather than position control. *Exp. Brain Res.*, 62(3) :451–458, 1986.

[20] N. Burgess. Spatial cognition and the brain. *Annals of the New York Academy of Sciences*, 1124(1) :77–97, 2008.

[21] X. Cao, J. J. Li, and R. Balakrishnan. Peephole pointing : Modeling acquisition of dynamically revealed targets. In *Proc. CHI '08*, pages 1699–1709. ACM, 2008.

[22] S. K. Card, T. P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23 :396–410, July 1980.

[23] S. K. Card, M. Pavel, and J. E. Farrell. Window-based computer dialogues. In *Proc. INTERACT '04*, pages 239–243, 1984.

[24] O. Chapuis. Gestion des fenêtres : enregistrement et visualisation de l'interaction. In *Proc. IHM '05*, pages 255–258. ACM, 2005.

[25] O. Chapuis, R. Blanch, and M. Beaudouin-Lafon. Fitts' law in the wild : A field study of aimed movements. Technical Report 1480, LRI, Univ. Paris-Sud, France, December 2007. 11 pages.

[26] O. Chapuis and N. Roussel. Metisse is not a 3D desktop ! In *Proc. UIST'05*, pages 13–22. ACM, 2005.

[27] O. Chapuis and N. Roussel. Copy-and-paste between overlapping windows. In *Proc. CHI'07*, pages 201–210. ACM, 2007.

[28] E. Clarke. Rhythm and timing in music. In *The Psychology of Music*, pages 473–500. Academic Press, 1999.

[29] A. Cockburn, C. Gutwin, and S. Greenberg. A predictive model of menu performance. In *Proc. CHI '07*, pages 627–636. ACM, 2007.

[30] F. I. Craik and R. S. Lockhart. Levels of processing : A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*, 11(6) :671–684, 1972.

[31] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proc. CHI '04*, pages 175–182. ACM, 2004.

[32] P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. UIST '04*, pages 193–196. ACM, 2004.

[33] B. Dresp, S. Durand, and S. Grossberg. Depth perception from pairs of overlqpping cues in pictorial displays. *Spatial Vision*, 15 :255–276, 2002.

[34] G. Faure. Mémoire spatiale & gestion de fenêtres. In *Rencontre Doctoral IHM '09*, 2009. 4 pages.

[35] G. Faure, O. Chapuis, and M. Beaudouin-Lafon. Acquisition of animated and pop-up targets. In *Proc. INTERACT '09*, pages 372–385. Springer and IFIP, 2009.

[36] G. Faure, O. Chapuis, and M. Beaudouin-Lafon. Perception de la profondeur en gestion des fenêtres. In *Proc. IHM '10*, pages 149–152. ACM, 2010.

[37] G. Faure, O. Chapuis, and N. Roussel. Power tools for copying and moving : Useful stuff for your desktop. In *Proc. CHI '09*, pages 1675–1678. ACM, 2009.

[38] J.-D. Fekete, N. Elmqvist, and Y. Guiard. Motion-pointing : target selection using elliptical motions. In *Proc. CHI '09*, pages 289–298. ACM, 2009.

[39] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47 :381–391, 1954.

[40] T. Flash and E. Henis. Arm trajectory modifications during reaching towards visual targets. *J. Cognitive Neuroscience*, 3(3) :220–230, 1991.

[41] C. Forlines, D. Vogel, and R. Balakrishnan. Hybridpointing : fluid switching between absolute and relative pointing with a direct input device. In *Proc. UIST'06*, pages 211–220. ACM, 2006.

[42] P. Fraisse. Rhythm and tempo. In *The Psychology of Music*, pages 149–180. Academic Press, 1982.

[43] E. Ghomi, G. Faure, S. Huot, O. Chapuis, and M. Beaudouin-Lafon. Using rhythmic patterns as an input method. In *Proc. CHI '12*. ACM, 2012. 10 pages, to appear.

[44] L. Glass. Synchronization and rhythmic processes in physiology. *Nature*, 410(6825) :277–284, 2001.

[45] C. Gonzalez. Does animation in user interfaces improve decision making ? In *Proc. CHI '96*, pages 27–34. ACM, 1996.

[46] S. Grossberg. Cortical dynamics of three-dimensional figure–ground perception of two-dimensional pictures. *Psychological Review*, 104(3) :618–658, 1997.

[47] T. Grossman, P. Dragicevic, and R. Balakrishnan. Strategies for accelerating on-line learning of hotkeys. In *Proc. CHI '07*, pages 1591–1600. ACM, 2007.

[48] S. Guedana. *Conception, mise en oeuvre et évaluation de systèmes de communication multi-échelles*. PhD thesis, Université Paris Sud XI, 2009.

[49] C. R. C. Guibal and B. Dresp. Interaction of color and geometric cues in depth perception : When does "red" mean "near" ? *Psychological Research*, 69 :30–40, 2004.

[50] K. Hasan, T. Grossman, and P. Irani. Comet and target ghost : techniques for selecting moving targets. In *Proc. CHI '11*, pages 839–848. ACM, 2011.

[51] D. A. Henderson, Jr. and S. Card. Rooms : the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph.*, 5(3) :211–243, 1986.

[52] W. Hick. On the rate of gain of information. *Quart. J. Exper. Psych.*, 4 :11–36, 1952.

[53] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proc. CHI'05*, pages 451–460. ACM, 2005.

[54] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proc. CHI'05*, pages 451–460. ACM, 2005.

[55] K. Hinckley, F. Guimbretiere, P. Baudisch, R. Sarin, M. Agrawala, and E. Cutrell. The springboard : multiple modes in one spring-loaded control. In *Proc. CHI '06*, pages 181–190. ACM, 2006.

[56] E. R. Hoffmann. Capture of moving targets : a modification of Fitts' law. *Ergonomics*, 34 :211–220, 1991.

[57] A. J. Hornof and D. E. Kieras. Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. In *Proc. CHI '99*, pages 410–417. ACM, 1999.

[58] D. R. Hutchings, G. Smith, B. Meyers, M. Czerwinski, and G. Robertson. Display space usage and window management operation comparisons between single monitor and multiple monitor users. In *Proc. AVI '04*, pages 32–39. ACM, 2004.

[59] D. R. Hutchings and J. Stasko. Quickspace : new operations for the desktop metaphor. In *Proc. CHI '02*, pages 802–803. ACM, 2002.

[60] D. R. Hutchings and J. Stasko. Revisiting display space management : understanding current practice to inform next-generation design. In *Proc. GI '04*, pages 127–134. Canadian Hum.-Comp. Comm. Soc., 2004.

[61] R. Hyman. Stimulus information as a determinant of reaction time. *J. Exper. Psych.*, 45 :188–196, 1953.

[62] E. W. Ishak and S. K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *Proc. UIST '04*, pages 189–192. ACM, 2004.

[63] M. Jeannerod. *The neural and behavioural organization of goal directed movements*. Clarendon Press, Oxford, 1988.

[64] J. Johnson, T. L. Roberts, D. C. Smith, C. H. Irby, M. Beard, and K. Mackey. The xerox star : A retrospective. *IEEE Computer*, 22(9) :11–29, 1989.

[65] E. Kandogan and B. Shneiderman. Elastic windows : improved spatial layout and rapid multiple window operations. In *Proc. AVI '96*, pages 29–38. ACM, 1996.

[66] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm : analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7) :881–892, Jully 2002.

[67] V. Kaptelinin. Umea : translating interaction histories into project contexts. In *Proc. CHI '03*, pages 353–360. ACM, 2003.

[68] A. Kay. *The Reactive Engine*. PhD thesis, University of Utah, USA, 1969.

[69] D. Kieras. Using the keystroke-level model to estimate execution times. University of Michigan, 1993, 2001.

[70] M. Kobayashi and T. Igarashi. Boomerang : suspendable drag-and-drop interactions based on a throw-and-catch metaphor. In *Proc. UIST'07*, pages 187–190. ACM, 2007.

[71] G. Kurtenbach and W. Buxton. Issues in combining marking and direct manipulation techniques. In *Proc. UIST '91*, pages 137–144. ACM, 1991.

[72] G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proc. CHI '94*, pages 258–264. ACM, 1994.

[73] V. Lantz and R. Murray-Smith. Rhythmic interaction with a mobile device. In *Proc. NordiCHI '04*, pages 97–100. ACM, 2004.

[74] H. G. MacDougall and S. T. Moore. Marching to the beat of the same drummer : the spontaneous tempo of human locomotion. *Journal of Applied Physiology*, 99, 2005.

[75] W. E. Mackay, C. Appert, M. Beaudouin-Lafon, O. Chapuis, Y. Du, J.-D. Fekete, and Y. Guiard. Touchstone : exploratory design of experiments. In *Proc. CHI '07*. ACM, 2007.

[76] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7 :91–139, 1992.

[77] S. Malacria, E. Lecolinet, and Y. Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces : the cyclostar approach. In *Proc. CHI '10*, pages 2615–2624. ACM, 2010.

[78] R. Manuel. *Histoire de la Musique*. Encyclopédie de la Pléiade, Gallimard, 1960.

[79] S. Maury, S. Athénes, and S. Chatty. Rhythmic menus : toward interaction based on rhythm. In *Proc. CHI '99 EA*, pages 254–255. ACM, 1999.

[80] D. Meyer, J. Smith, S. Kornblum, R. Abrams, and C. Wright. Optimality in human motor performance : Ideal control of rapid aimed movements. *Psych. Review*, 95 :340–370, 1988.

[81] R. C. Miller and B. A. Myers. Synchronizing clipboards of multiple computers. In *Proc. UIST '99*, pages 65–66. ACM, 1999.

[82] R. C. Miller and B. A. Myers. Multiple selections in smart text editing. In *Proc. IUI '02*, pages 103–110. ACM, 2002.

[83] D. Moelants. Preferred tempo reconsidered. In *Proc. Music Perception and Cognition*, 2002.

[84] D. Mould and C. Gutwin. The effects of feedback on targeting with multiple moving targets. In *Proc. GI '04*, pages 25–32. Canadian Hum.-Comp. Comm. Soc., 2004.

[85] B. Myers. A taxonomy of window manager user interfaces. *IEEE Computer Graphics and Applications*, 8(5) :65–84, 1988.

[86] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited : Implications of a long-term click-stream study of browser usage. In *Proc. CHI '07*, pages 597 – 606. ACM, 2007.

[87] N. Oliver, M. Czerwinski, G. Smith, and K. Roomp. Relalttab : assisting users in switching windows. In *Proc. IUI '08*, pages 385–388. ACM, 2008.

[88] N. Oliver, G. Smith, C. Thakkar, and A. C. Surendran. Swish : semantic analysis of window titles and switching history. In *Proc. IUI '06*, pages 194–201. ACM, 2006.

[89] G. S. Patrick, P. Baudisch, G. Robertson, M. Czerwinski, B. Meyers, D. Robbins, and D. Andrews. Groupbar : The taskbar evolved. In *Proc. OZCHI '03*, pages 34–43, 2003.

[90] L. A. Petitto, S. Holowka, L. E. Sergio, B. Levy, and D. J. Ostry. Baby hands that move to the rhythm of language : hearing babies acquiring sign languages babble silently on the hands. *Cognition*, 93(1) :43 – 73, 2004.

[91] R. Plamondon and A. Alimi. Speed/accuracy trade-offs in target-directed movements. *Behavioral and Brain Sciences*, 20(2) :279–349, 1997.

[92] T. Rammsayer and S. Lima. Duration discrimination of filled and empty auditory intervals : Cognitive and perceptual factors. *Perception and Psychophysics*, 50 :565–574, 1991.

[93] A. Raskin. Firefox panorama : Tab candy evolved. http ://www.azarask.in/blog/post/designing-tab-candy/.

[94] P. Ravasio, S. G. Schär, and H. Krueger. In pursuit of desktop evolution : User problems and practices with modern desktop systems. *ACM ToCHI*, 11(2) :156–180, 2004.

[95] J. Rekimoto. Pick-and-drop : a direct manipulation technique for multiple computer environments. In *Proc. UIST '97*, pages 31–39. ACM, 1997.

[96] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich. Data mountain : using spatial memory for document management. In *Proc. UIST '98*, pages 153–162. ACM, 1998.

[97] G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. R. Hutchings, B. Meyers, D. Robbins, and G. Smith. Scalable fabric : flexible task management. In *Proc. AVI '04*, pages 85–89. ACM, 2004.

[98] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery : a 3d window manager. In *Proc. CHI '00*, pages 494–501. ACM, 2000.

[99] O. Sacks. *Musicophilia : Tales of Music and the Brain*, volume 1. Vintage Books, 2 edition, 2008.

[100] C. Schlienger, S. Conversy, S. Chatty, M. Anquetil, and C. Mertz. Improving users' comprehension of changes with animation and sound : An empirical assessment. In *Proc. INTERACT '07*, pages 207–220. LNCS 4662, Springer, 2007.

[101] C. Schlienger, P. Dragicevic, C. Ollagnon, and S. Chatty. Les transitions visuelles différenciées : principes et applications. In *Proc. IHM '06*, pages 59–66. ACM, 2006.

[102] J. Scott, D. Dearman, K. Yatani, and K. N. Truong. Sensing foot gestures from the pocket. In *Proc. UIST*, pages 199–208, 2010.

[103] S. Seow. Information theoretic models of HCI : A comparison of the Hick-Hyman law and Fitts' law. *HCI*, 20(3) :315–352, 2005.

[104] E. Sharlin, B. Watson, Y. Kitamura, F. Kishino, and Y. Itoh. On tangible user interfaces, humans and spatiality. *Personal Ubiquitous Comput.*, 8 :338–346, September 2004.

[105] E. Sharlin, B. Watson, Y. Kitamura, F. Kishino, and Y. Itoh. On tangible user interfaces, humans and spatiality. *Personal and Ubiquitous Computing*, 8 :338–346, 2004.

[106] B. Shneiderman. Direct manipulation : a step beyond programming languages. *IEEE Computer*, 16(8) :57–69, 1983.

[107] B. Shneiderman. Creating creativity : user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction*, 7(1) :114–138, 2000.

[108] R. B. Smith and A. Taivalsaari. Generalized and stationary scrolling. In *Proc. UIST'99*, pages 1–9. ACM, 1999.

[109] D. Stewart and S. Sjanssen. Xmonad. In *Proc. Haskell '07*, pages 119–119. ACM, 2007.

[110] W. Stuerzlinger, O. Chapuis, D. Phillips, and N. Roussel. User interface façades : towards fully adaptable user interfaces. In *Proc. UIST '06*, pages 309–318. ACM, 2006.

[111] I. Sutherland. *Sketchpad, a man-machine graphical communication system*. PhD thesis, Massachusetts Institute of Technology, USA, January 1963.

[112] I. Sutherland. Sketchpad a man-machine graphical communication system. In *25 years of DAC : Papers on Twenty-five years of electronic design automation*, pages 507–524. ACM, 1988.

[113] C. Szentgyorgyi and E. Lank. Five-key text input using rhythmic mappings. In *Proc. ICMI '07*, pages 118–121. ACM, 2007.

[114] A. Tabard, W. Mackay, N. Roussel, and C. Letondal. Pagelinker : integrating contextual bookmarks within a browser. In *Proc. CHI '07*, pages 337–346. ACM, 2007.

[115] S. Tak, A. Cockburn, K. Humm, D. Ahlström, C. Gutwin, and J. Scarr. Improving window switching interfaces. In *Proc. INTERACT '09*, pages 187–200. Springer-Verlag, 2009.

[116] C. Tashman. Windowscape : a task oriented window manager. In *Proc. UIST'06*, pages 77–80, 2006.

[117] M. Waldner, M. Steinberger, R. Grasset, and D. Schmalstieg. Importance-driven compositing window management. In *Proc. CHI '11*, pages 959–968. ACM, 2011.

[118] T. Westeyn and T. Starner. Recognizing song-based blink patterns : Applications for restricted and universal access. In *Proc. FGR '04*, pages 717–722. IEEE, 2004.

[119] J. O. Wobbrock. Tapsongs : tapping rhythm-based passwords on a single binary sensor. In *Proc. UIST '09*, pages 93–96. ACM, 2009.

[120] Q. Xu and G. Casiez. Push-and-pull switching : window switching based on window overlapping. In *Proc. CHI '10*, pages 1335–1338. ACM, 2010.

[121] S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard. Human on-line response to target expansion. In *Proc. CHI '03*, pages 177–184. ACM, 2003.