ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2015-ENST-0079

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Quentin Roy

le 11 décembre 2015

Techniques d'interaction pour applications riches sur tablettes multi tactiles

Directeur de thèse : **Yves GUIARD**
Directeur de thèse : **Éric LECOLINET**

Jury

M. Stéphane HUOT, Directeur de recherches, Inria Lille Nord-Europe	Rapporteur
M. Stéphane CONVERSY, Maître de Conférences HDR, ENAC	Rapporteur
M. Michel BEAUDOUIN-LAFON, Professeur, LRI	Examinateur
M. Laurent LAUNAY, Senior Software Manager, GE HealthCare	Examinateur

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

THÈSE

Abstract

This PhD in collaboration with *GE HealthCare* aims at improving interaction on multitouch tablets so as to allow these devices to support applications with a large number of functionalities. The context of the research was digital radiology, an especially difficult task on small surfaces because it requires precise manipulations and the use of numerous and complex tools. The design and the development of a medical imagery software prototype confronted us with problems of the real world. We tried to provide concrete and innovative solutions to these problems.

The first part of this research aimed at optimizing the use of extant input channels of multitouch tablets by designing and experimentally evaluating a number of new interaction techniques. In comparison with traditional techniques on tablets (menubars and toolbars), the *Sigma-Menu* and the *Finger-Menu* reduce substantially the consumption of screen real estate, a scarce resource on tablets, at essentially no performance cost. The *Augmented Letters* technique leverages the linguistic capabilities of users to substantially increase the number of recalled shortcuts. Finally, the *Multitouch Menubar*, a technique that is backward compatible with traditional menubars, increases the input information by taking into account the number of contacts on the surface. Most of these techniques (*Sigma-Menu*, *Finger-Menu*, *MultiTouch Menubar*) were implemented on our radiology prototype.

In the second part of this research we started from the observation that any contact is a dual-surface event. While the pointing coordinates tell the system which part of the screen surface is being touched, the identification of the pointing finger tells it which part of the body surface is touching the screen. These two pieces of information being essentially independent from each other, the standard target pointing task can be thought of as a double information channel in Shannon's sense. We report an experiment which confirmed that combining the identification of the contact point on the screen with the identification of the finger makes it possible, for a given amount of available screen real estate, not only to substantially expand the size of the command vocabularies (considered in this study as an entropy and measured in bits), but also to increase users' effective information throughput (measured in bits/s).

Résumé

L'objectif de cette thèse CIFRE en collaboration avec *GE HealthCare* est de contribuer à enrichir l'interaction sur tablettes tactiles en rendant utilisables sur ces surfaces réduites un plus grand nombre de fonctionnalités. Le contexte spécifique auquel nous avons songé en priorité est la revue radiologique numérique, une tâche particulièrement difficile à implémenter sur des surfaces réduites parce qu'elle requiert des manipulations précises et l'emploi d'outils multiples et complexes. La conception et le développement d'un prototype destiné à une application d'imagerie médicale a eu pour effet de nous confronter à des problèmes réels auxquels nous avons tenté d'apporter des solutions concrètes.

Dans le premier volet de cette thèse, d'inspiration appliquée, nous nous sommes efforcés d'optimiser l'exploitation des canaux d'entrées existants sur tablette en concevant et en évaluant expérimentalement un certain nombre de techniques d'interaction nouvelles. Ainsi le *Sigma-Menu* et le *Finger-Menu* réduisent considérablement la consommation de l'espace-écran, ressource rare sur tablette, pour des performances similaires à celles obtenues avec les techniques traditionnelles sur tablette (barre d'outils et barre de menus). La technique *Augmented Letters* tire parti des capacités langagières des utilisateurs pour accroître substantiellement le nombre de raccourcis gestuels mémorisables. Enfin la *Barre d'Outils Multitouch*, une technique rétro-compatible avec les barres de menus traditionnelles, accroît l'information d'entrée en prenant en considération le nombre des points de contact. La plupart de ces techniques (*Sigma-Menu*, *Finger-Menu*, *Multi-Touch Menubar*) ont été intégrées à notre prototype de radiologie.

Dans le second volet de cette recherche, d'inspiration plus fondamentale, nous sommes partis du constat que tout contact est un événement bi-surface. Si les coordonnées du pointage disent au système quelle partie de la surface de l'écran a été touchée, l'identification du doigt qui pointe peut lui permettre de savoir quelle partie de la surface du corps est venue toucher l'écran. Ces deux informations étant en pratique largement indépendantes, on peut voir dans le pointage de cible classique un double canal d'information au sens de Shannon. Nous confirmons avec une expérience qu'en combinant l'identification du point de contact sur l'écran avec l'identification du doigt il devient possible, à un niveau donné de surface d'écran, non seulement d'étendre substantiellement la taille des vocabulaires de commandes (considérée dans notre étude comme une entropie et mesurée en bits) mais également d'accroître les débits effectifs d'information (mesurés en bits/s).

Remerciements

Je tiens, tout d'abord, à remercier mes deux directeurs de thèse, Éric Lecolinet et Yves Guiard. Ils m'ont accompagné pendant ces trois années et m'ont offert un guidage précieux dans mon apprentissage de la recherche. Je remercie également Laurent Launay et François Peter pour leur encadrement, leur support et leurs conseils.

En second lieu, je voudrais remercier Stéphane Huot, rapporteur de ma thèse mais aussi celui qui, en premier, m'a donné goût à la recherche et à l'Interaction Humain-Machine.

Je remercie l'ensemble des membres de mon jury pour avoir lu mes travaux, pour leurs remarques constructives et écoute durant ma soutenance.

Merci aux membres de l'équipe VIA qui ont partagé avec moi de nombreux hauts et bas : Simon Perrault, Dong-Bach Vo, Marcos Serrano, Hind Gacem, Julie Wagner, Aurélie Cohé, Thibaut Jacob, Marc-Emmanuel Perrin et Gilles Bailly.

Merci aussi à tous les collègues et amis rencontrés à GE HealthCare. J'espère qu'ils me pardonneront de ne pas m'aventurer à tous les citer ici.

Un grand merci à tous mes amis qui m'ont soutenu et encouragé.

Merci à tous ceux qui ont participé à mes expériences parfois longues et souvent fastidieuses.

Mille mercis à mes parents pour m'avoir toujours encouragé à développer un esprit critique, à rester curieux et pour m'avoir porté jusqu'ici.

Et pour finir, merci à Lucie, qui malgré cette aventure dans laquelle elle s'est trouvée embarquée un peu malgré elle, est toujours restée positive, encourageante, et a même trouvé le courage de m'épouser en cours de route.

Table des matières

Liste des figures

Liste des tableaux

1 Introduction

Le cadre d'étude de cette thèse CIFRE, réalisée avec *GE HealthCare*, est la visualisation interactive des données radiologiques, souvent appelée « revue radiologique » par les spécialistes, sur tablette tactile. La revue radiologique numérique est une tâche difficile nécessitant l'utilisation et la manipulation d'outils complexes et une grande précision. Ces outils, traditionnellement disponibles sur ordinateur de bureau (clavier, souris, un voire souvent deux écrans de grande taille), sont actuellement portés sur tablettes tactiles pour permettre aux radiologues d'accéder à leurs dossiers lors de leurs déplacements. Cependant, les tablettes tactiles posent de nombreux problèmes d'interaction : occultation, imprécision, absence de clavier physique, etc. Toutes ces limitations réduisent l'expressivité des applications développées pour tablettes et compliquent, notamment, la manipulation des nombreux outils et modes nécessaires à la revue radiologique. Cette thèse s'inscrit dans cette problématique et vise à proposer des solutions pour enrichir l'interaction sur les surfaces interactives en prenant comme contexte l'analyse radiologique.

Le partenaire industriel de cette thèse, *GE Healthcare*, est une filiale de *General Electric* qui construit notamment des technologies d'imagerie médicale, telles que des scanners ou des stations d'IRM, ainsi que les suites logicielles permettant de travailler sur les images produites. Dans le cadre de la thèse, on s'intéresse plus particulièrement à *Volume Viewer*, une plateforme de traitement, de visualisation et d'analyse d'imagerie médicale (voir Figure 1.1).

La conception et le développement d'un prototype destiné à une application d'imagerie médicale a eu pour effet de nous confronter à des problèmes réels auxquels nous avons tenté d'apporter des solutions concrètes.

Dans cette introduction, nous nous intéressons dans un premier temps aux objectifs et aux besoins de l'analyse radiologique en matière d'interaction utilisateur. Nous décrivons ensuite un scénario classique de l'utilisation d'un logiciel d'analyse radiologique puis détaillons les différents cas où l'introduction d'une tablette tactile fait sens pour un radiologue.

1.1 La revue radiologique

La médecine moderne repose de plus en plus sur l'imagerie médicale et la radiologie, d'où des contraintes de productivité de plus en plus fortes sur les radiologues.

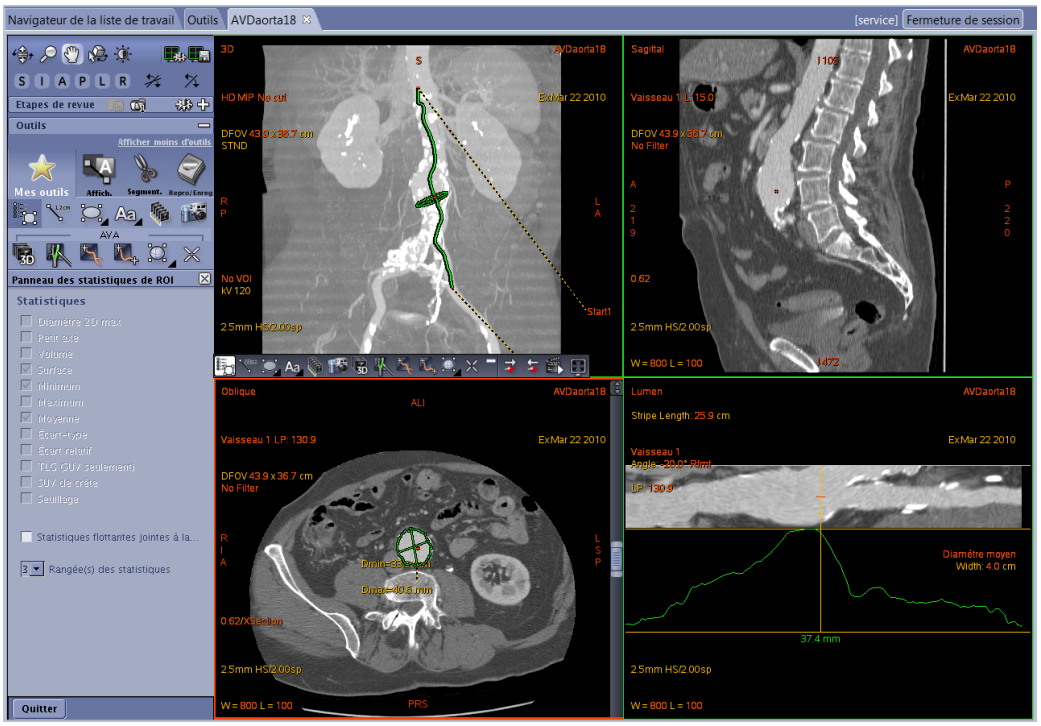


Figure 1.1 – Interface de *Volume Viewer 6* et segmentation automatique d'un vaisseau sanguin. De gauche à droite et de haut en bas : Vue 3D, coupe sagittal, coupe axial et coupe « lumen ». Le vaisseau sanguin segmenté apparait en vert. La coupe « lumen » suit le vaisseau sanguin (celui-ci apparait donc aplati).

Le but principal de leur travail est de détecter, localiser, observer et mesurer des artefacts anormaux à l'intérieur de l'organisme d'un patient, par exemple un anévrisme, une tumeur, une fracture osseuse, etc. Pour ce faire, les radiologues exploitent des technologies capables de produire une image du corps d'une personne qui est aujourd'hui souvent en trois dimensions. De plus, l'évolution au cours du temps est parfois disponible.

Il existe de nombreuses spécialités en radiologie. Chacune est en général spécialisée pour un certain type de pathologies, par exemple les cancers, les problèmes respiratoires ou les maladies vasculaires (Figure 1.1). Chaque spécialité a un protocole analytique différent pour lequel des outils spécifiques sont nécessaires.

Fonctionnalités principales. Parmi les fonctionnalités communes les plus utilisées par l'ensemble des différentes spécialités, on peut citer :

— le parcours sur les 6 dimensions de l'intérieur d'une capture 3D du corps d'un

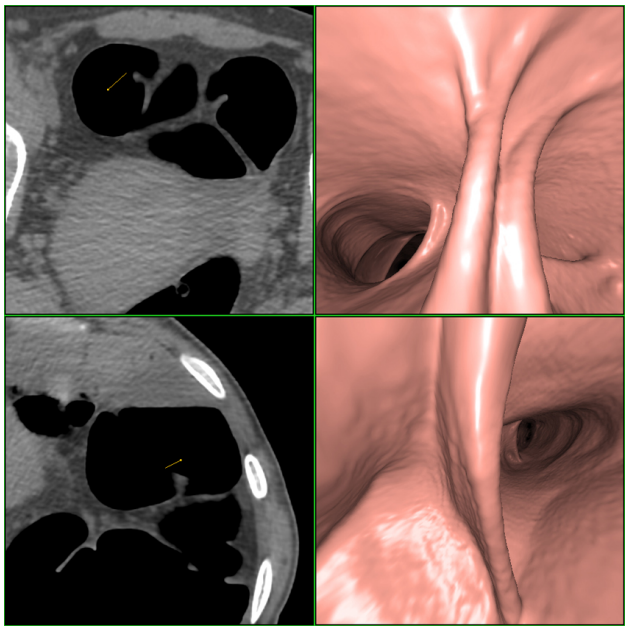


Figure 1.2 – A droite : rendu volumique de l'intérieur d'un vaisseau sanguin. À gauche : coupes du volume.

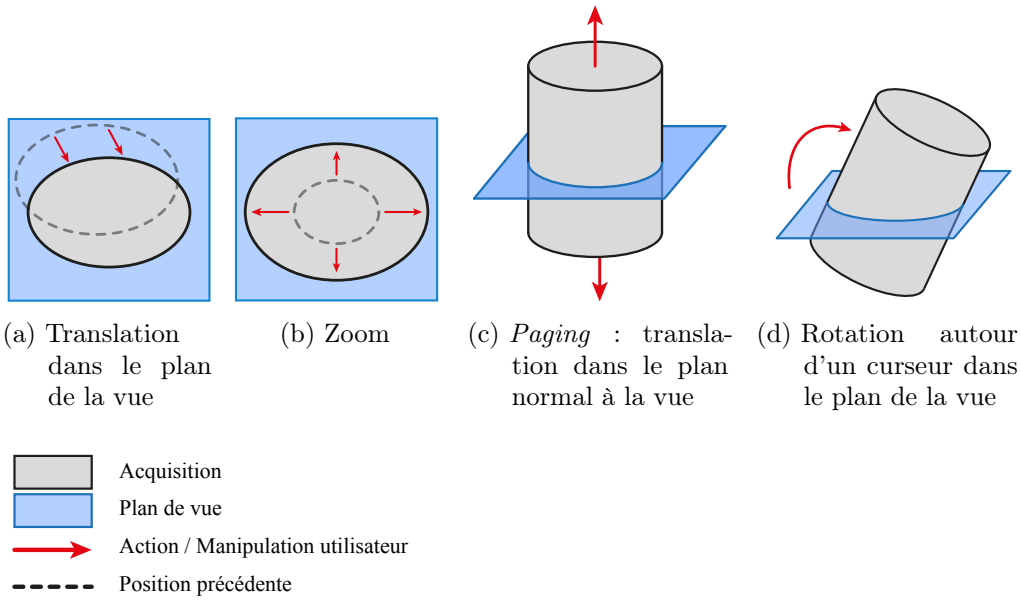
patient (translations et rotations, voir la Figure 1.3).

— l'utilisation de différentes vues ayant leurs spécificités propres en termes d'interaction et d'outils disponibles (projection d'intensité maximale ou minimale, rendu volumique, plan courbe), etc.

— l'ajustement de divers paramètres tels que le contraste, la luminosité ou l'épaisseur de la coupe visualisée.

— différentes prises de mesures (longueur, distance, aire, volume, etc.)

Visualisation par coupe. Il est important de noter que bien qu'il soit possible d'effectuer un rendu volumique de l'image 3D, un radiologue travaille essentiellement sur une visualisation par coupe. Ainsi, au lieu d'observer le volume depuis un point de vue extérieur, l'utilisateur obtient un plan interne au volume lui permettant donc d'en étudier l'intérieur. La Figure 1.2 montre la différence entre une vue par coupe et un rendu volumique. Il également possible de travailler avec des coupes non planaires. Par exemple, la vue en bas à droite de la Figure 1.1 suit un vaisseau sanguin (la segmentation de celui-ci apparaît en vert sur les deux vues de gauche). Les courbures du vaisseau apparaissent alors « aplaties » ce qui permet, notamment, de mieux observer l'évolution de son épaisseur sur toute sa longueur.



(a) Translation dans le plan de la vue

(b) Zoom

(c) *Paging* : translation dans le plan normal à la vue

(d) Rotation autour d'un curseur dans le plan de la vue

Figure 1.3 – Navigation au sein d'une image 3D avec *Volume Viewer*

1.2 Scénario

Afin de guider les fonctionnalités à implémenter au sein du prototype ainsi que le type de tâche qui nous intéresse, nous avons défini un scénario d'utilisation précis en nous reposant sur un protocole d'analyse radiologique réel.

1. Le radiologue règle le contraste et la luminosité de la capture afin d'optimiser le rendu visuel.
2. Il recentre le volume et l'agrandit pour l'adapter à la zone d'affichage (Figures 1.3a et 1.3b).
3. Il parcourt longitudinalement l'ensemble de la capture dans l'objectif de repérer des artefacts (Figure 1.3c). Il identifie un anévrisme [1] non loin du cœur sur l'artère descendante antérieure gauche.
4. Il tourne alors le volume afin de se positionner dans le plan de l'anévrisme (Figure 1.3d).
5. Il utilise la fonctionnalité de segmentation automatique des vaisseaux sanguins et ajuste le résultat en repositionnant précisément la ligne passant au centre du vaisseau sanguin au niveau de l'anévrisme (Figure 1.4a).

1. Un anévrisme est une dilatation de la paroi d'un vaisseau sanguin aboutissant à la formation d'une poche de sang de taille variable. Le danger est celui d'une rupture ce qui entraîne une hémorragie avec un risque de mortalité.

6. Il positionne finalement une règle du centre du vaisseau sanguin jusqu'en haut de l'anévrisme afin de mesurer sa hauteur (Figure 1.4b).

(a) Repositionnement de la ligne centrale du vaisseau sanguin après une segmentation automatique

(b) Positionnement d'une règle

Figure 1.4 – Mesure d'un anévrisme

Un radiologue se sert habituellement d'un clavier, d'une souris à trois boutons ainsi que de deux écrans de 19 pouces permettant la visualisation de plus de 4 vues simultanément. Toutefois, la popularisation des tablettes tactiles (72% des praticiens nord-américains en utilisent une[2]) et les nouveaux usages médicaux (mobilité, démonstration aux patients et aux médecins, etc.) motivent aujourd'hui la recherche d'une solution adaptée à ce type de dispositifs.

À l'exception des deux dernières étapes, l'ensemble des fonctionnalités nécessaires de ce scénario ont été implémentées dans le prototype de radiologie durant la thèse.

1.3 Cas d'utilisation d'une tablette tactile pour la radiologie

L'objectif d'une application de radiologie numérique sur tablette n'est pas de remplacer à terme la station de travail habituelle. Il s'agit d'un dispositif complémentaire justifié par différents cas d'utilisation.

Mobilité. La pratique de la médecine moderne demande au radiologue de se déplacer fréquemment pour s'entretenir avec ses confrères ou ses patients. Les temps de transports sont aujourd'hui perdus. Une application mobile peut permettre à un radiologue d'en tirer parti, par exemple pour se remémorer un examen spécifique.

Entretiens. Après un examen, un radiologue effectue des captures d'écran mettant en avant les éléments importants et les mesures effectuées. Ces captures sont ensuite imprimées afin de pouvoir servir de support aux entretiens avec les médecins et les

2. Manhattan Research : Taking the Pulse® U.S. http://manhattanresearch.com/Products-and-Services/Physician/Taking-the-Pulse-U-S

patients. Ces captures étant par définition statiques, un aller-retour supplémentaire sera nécessaire si un détail est manquant, ce qui est à la fois source de pression et une potentielle perte de temps. Un outil de radiologie mobile et interactif peut à la fois permettre de faire ressortir des éléments pouvant éventuellement avoir été manqués tout en offrant une meilleure visualisation.

1.4 Applications riches et tablette tactile

Nous appelons « application riche » une application comportant un grand nombre de fonctionnalités. Du fait du nombre d'outils qu'elles intègrent, la conception de ce type d'applications pose problème sur tablette tactile.

Aujourd'hui, les applications pour tablettes sont souvent « minimalistes » comparativement aux versions destinées aux ordinateurs de bureau (donc munis d'un clavier et d'une souris ou un *trackpad*). En terme de comparaison, le célèbre logiciel de traitement d'image Adobe Photoshop [3] est un bon exemple. Sur ordinateur de bureau, selon notre estimation, Photoshop comporte environ 2800 fonctionnalités différentes [4] (Figure 1.5a). Par contre, Photoshop Express [5], sa version mobile sur iPad, comporte moins de 50 fonctionnalités (Figure 1.5b), soit presque 60 fois moins.

On considère souvent la tablette comme une plateforme de consultation, mais pas de production [74]. Des études récentes ont montré que seulement 26% des utilisateurs de tablettes les considèrent comme efficaces pour la production [6], que seulement 15% du temps passé sur une tablette est utilisé pour créer du contenu [7] et que seulement 12% des étudiants disposant d'une tablette s'en servent pour créer des présentations ou des documents [8] [74]. Ces résultats coïncident avec les constatations faites lors de la conception de notre prototype de radiologie.

3. http://www.photoshop.com/products/photoshop

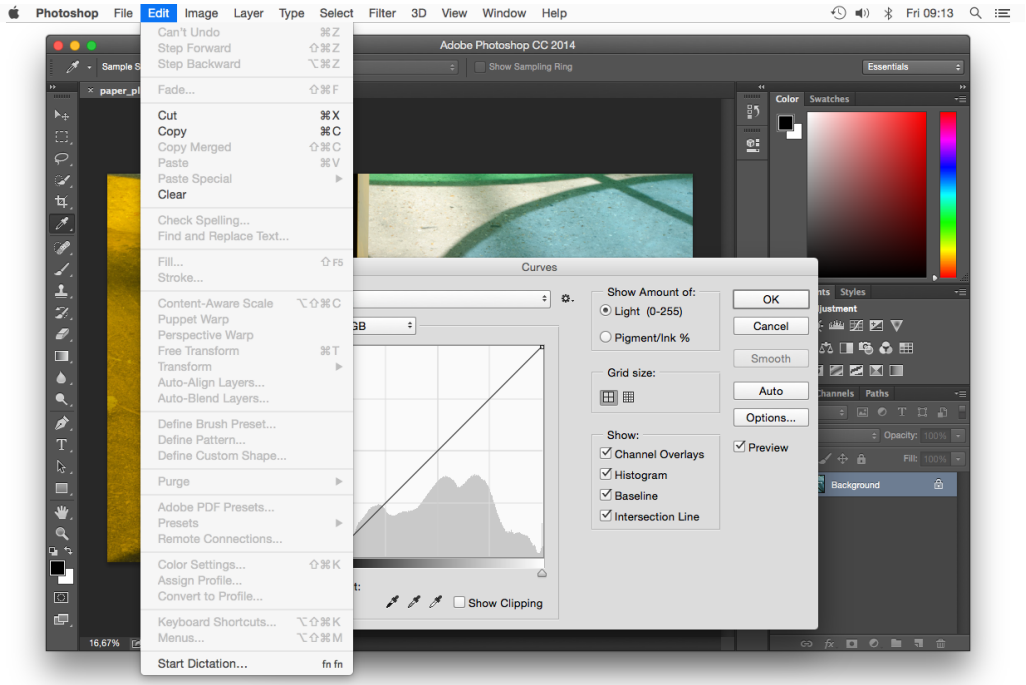
4. En utilisant MenuInspector [20], nous avons comptabilisé 648 *items* dans les menus de la version Mac OS X d'Adobe Photoshop CS6 (v. 13.0.6). Parmi ces *items*, 248 se terminent par trois points de suspension (« ... ») marquant la présence d'une boîte de dialogue. En nous basant sur un échantillon de 20 boîtes de dialogue ouvertes aléatoirement dans l'application, nous avons en moyenne compté 9,8 fonctionnalités différentes ($\sigma = 6{,}8$). D'où $648 + 248 \times (9{,}8 - 1) = 2830{,}4$ fonctionnalités présentes dans le logiciel. Ces fonctionnalités peuvent se présenter sous la forme de boutons, de champs textes, d'explorateurs de fichiers, de potentiomètre (*sliders*), de graphes de couleurs, etc (voir Figure 1.5a). Les boutons de validation ou d'annulation présents dans la grande majorité des boîtes de dialogues ne sont pas pris en compte dans le calcul. De même, certains champs associés (un champ texte couplé à un potentiomètre par exemple) ne furent comptabilisés que comme une seule fonctionnalité.

5. http://www.photoshop.com/products/photoshopexpress

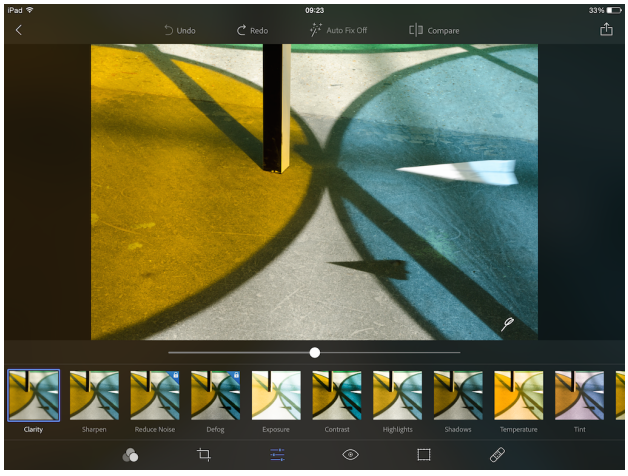
6. APQC survey : Personal Productivity in the Mobile Age. https://www.apqc.org/knowledge-base/documents/should-we-just-play-chopsticks-personal-productivity-mobile-age

7. Gartner Survey Says Entertainment Accounts for Half of Device Screen Time. http://www.gartner.com/newsroom/id/2590715

8. Nielsen survey : A computer in every classroom and a tablet in every backpack ?



(a) Adobe Photoshop sur ordinateur traditionnel



(b) Adobe Photoshop sur tablette tactile

Figure 1.5 – Adobe Photoshop

1.5 Organisation du mémoire

Le chapitre 2 est consacré à la caractérisation des tablettes tactiles et de leurs capacités interactives et à l'étude de leurs forces et faiblesses afin d'identifier des axes d'amélioration potentiels. Deux axes de travail se dégagent :

1. mieux exploiter les capacités en entrée existante grâce à la conception de nouvelles techniques d'interaction plus performantes. Nous nous sommes en particulier intéressés à la sélection et la transition entre différents modes de contrôle continu.
2. étendre les capacités en entrée en étudiant de nouveaux canaux d'entrée.

Le chapitre 3 a trait à la définition de la notion de mode, et à l'étude des différents types de délimiteurs permettant de basculer d'un mode à un autre. Nous y concluons que les modes sont omniprésents dans les interfaces informatiques et que le défi d'une application n'est pas d'en limiter l'usage, mais d'en maîtriser soigneusement les transitions, c'est-à-dire définir des délimiteurs efficaces et appropriés.

Le chapitre 4 est dédié à la conception de deux nouvelles techniques de transition entre modes en s'appuyant sur le modèle des modes et la caractérisation des délimiteurs présentés dans le chapitre précédent :

1. le *FingerMenu* qui associe à chaque doigt un outil, et
2. le *SigmaMenu* qui est activé grâce à un mouvement circulaire.

Ces deux techniques d'interaction se révélèrent aussi performantes que les techniques standard sur tablette (barre d'outils et de menus) tout en ne consommant aucun espace-écran.

Le chapitre 5 est consacré à la conception de deux autres nouvelles techniques d'interaction qui se concentrent plus particulièrement sur le support des utilisateurs experts :

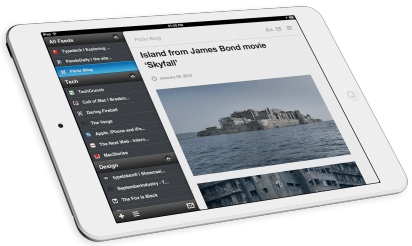
1. Les Lettres Augmentées qui s'appuient sur le langage pour permettre l'accès à un grand nombre de commandes facilement mémorisables. Notre expérience a montré que cette technique permettait environ 25% de mémorisation supplémentaire sans pour autant impacter la vitesse d'exécution.
2. La Barre de Menus *MultiTouch* qui s'inspire des barres de menu traditionnelles, mais permet une sélection plus rapide en fonction du nombre de doigts.

Le dernier chapitre (chapitre 6) est dédié à l'étude d'un nouveau canal d'interaction tenant compte non seulement des coordonnées d'un point de contact sur la

http://www.nielsen.com/us/en/insights/news/2013/a-computer-in-every-classroom-and-a-tablet-in-every-backpack.html

surface tactile de la tablette, mais également des coordonnées sur la surface de la peau de l'utilisateur grâce à l'identification des différents doigts. Nous montrons théoriquement puis expérimentalement que l'exploitation de ce canal, que nous appelons verre+peau, permet une nette amélioration des performances avec un grand nombre de commandes différentes. Pour ce faire nous nous appuyons sur une nouvelle métrique expérimentale en IHM : le débit de Shannon [200].

2 Tablette à écran tactile : un ordinateur épuré, mais limité



(a) Apple iPad

(b) Samsung Galaxy Tab

(c) Microsoft Surface Pro

Figure 2.1 – Exemples de tablettes à écran multi-tactile

Officiellement, une tablette tactile ou une ardoise numérique désigne un « ordinateur portable et ultraplat, qui se présente comme un écran tactile (sans clavier) et qui permet notamment d'accéder à des contenus multimédias » [63]. Depuis la mise sur le marché du premier iPad par Apple en 2010, les tablettes tactiles (Figure 2.1) se sont rapidement démocratisées auprès du grand public. Selon les termes de Steve Jobs, l'idée était de « mettre un ordinateur entier dans un livre que vous pourrez transporter et que vous apprendrez à utiliser en 20 minutes » [145].

En introduction (Section 1.4), nous avons vu que les tablettes actuelles étaient rarement utilisées pour l'édition ou la production de contenu et qu'il existait un écart important entre le nombre de fonctionnalités disponibles sur les applications pour ordinateur de bureau et les applications pour tablette. Plusieurs raisons peuvent expliquer cet écart. Tout d'abord, les usages d'une tablette diffèrent de ceux d'un

ordinateur traditionnel, et proposer une application minimaliste peut être un choix de conception. Certaines applications nécessitent néanmoins une large palette de fonctionnalités, par exemple le logiciel de radiologie 3D de GE HealthCare. Or, en pratique, il s'avère difficile d'intégrer toutes ces fonctionnalités sur tablette.

Ce chapitre vise à identifier les principaux freins à la conception d'applications riches, c'est-à-dire comportant plus d'une cinquantaine de fonctionnalités différentes, sur tablette tactile.

Il est toutefois difficile de couvrir l'ensemble des propriétés et paramètres entrant en jeu. Afin de réduire la couverture de notre espace, on se concentrera donc en particulier sur la sélection de commandes et le contrôle continu, une problématique identifiée lors de la conception de notre prototype de radiologie. On ignorera les questions liées à la visualisation et à l'entrée de texte et on se concentrera principalement sur l'interaction avec les doigts, sans stylet – un outil rarement utilisé aujourd'hui.

Dans un premier temps, nous nous intéresserons aux différents types de périphériques tactiles (Partie 2.1) ainsi que leurs propriétés intrinsèques. Nous étudierons ensuite le pointage sur tablette tactile (Partie 2.2), puis nous nous intéresserons à l'interaction gestuelle (Partie2.3).

2.1 Dispositifs d'entrée tactiles

Le terme tactile fait référence à la perception du toucher. Nous définissons un dispositif d'entrée tactile comme un type de périphérique capable de détecter le contact d'un autre objet, par exemple un doigt ou un stylet, sur sa surface [102].

Afin de mieux appréhender les propriétés spécifiques des tablettes, on commencera par positionner les dispositifs d'entrée tactiles parmi les autres dispositifs d'entrée. Nous parlerons par la suite des principales technologies tactiles (capacitives, résistives, optiques, etc.) puis de leurs propriétés (nombre de contact, forme du contact, détection de la pression, etc.). Nous nous intéresserons ensuite à l'intérêt et aux conséquences de la fusion du dispositif d'entrée tactile avec l'écran et, pour finir, nous évoquerons le problème du retour utilisateur.

2.1.1 Isotonique absolu

Il existe différentes classes de dispositifs d'entrée [81]. Les surfaces tactiles sont isotoniques et absolues.

Dispositifs d'entrée isométriques, élastiques ou isotoniques

Les périphériques isométriques mesurent la force qui leur est exercée. Ils opposent une résistance forte (quasi-infinie) au mouvement. C'est le cas des *trackpoints* [188] (Figure 2.2c).

(a) Souris

(b) *Trackball*

(c) *Trackpoint* [188]

(d) *Joystick*

(e) *Trackpad*

(f) Tablette Graphique

Figure 2.2 – Exemples de périphériques d'entrées

Les périphériques élastiques, sont des périphériques qui opposent au mouvement une résistance variable, notamment en fonction de la distance avec une position de d'équilbre. C'est le cas des *joysticks* (Figure 2.2d).

Enfin, les périphériques isotoniques exercent une résistance constante voire nulle au mouvement et mesurent typiquement le déplacement. Les tablettes tactiles, mais aussi les souris et les *Trackballs* appartiennent à cette classe (Figure 2.2).

Dispositifs d'entrée absolus ou relatifs

Les périphériques relatifs ne disposent d'aucun repère et mesurent uniquement une différence. Par exemple, une souris détecte son déplacement sur une surface. Si elle est soulevée, elle passe en état dit « débrayé » (*Out Of Range* ou *OOR*) [44] et n'est plus en mesure de capter ce mouvement (voir la Partie 2.2.2). Lorsqu'on la repose, même si sa position sur la surface a été modifiée, elle n'aura pas détecté de déplacement. Les *Trackballs* sont également des dispositifs relatifs, mais mesurent une rotation. Les périphériques relatifs permettent seulement un contrôle du curseur relatif (voir la Partie 2.2.1).

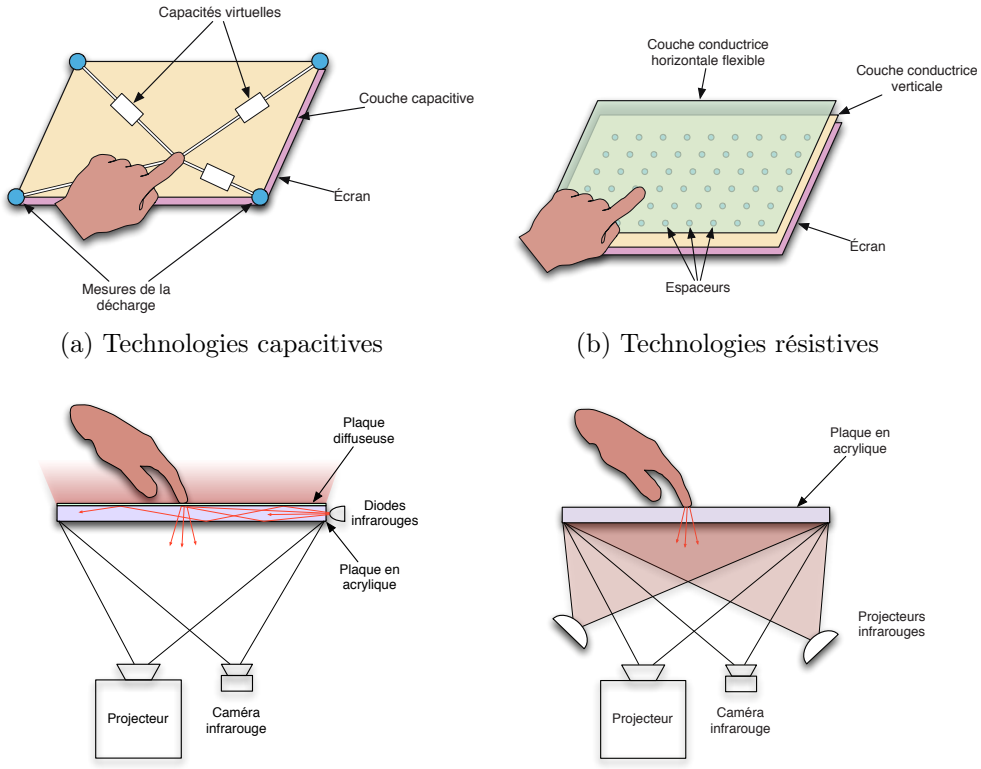
Les périphériques absolus sont des périphériques qui retournent des coordonnées dans un référentiel pourvu d'une origine. Les surfaces tactiles, comme les tablettes tactiles, les *trackpads* et les tablettes graphiques [1], sont des périphériques absolues

1. À ne pas confondre avec une tablette tactile, une tablette graphique (Figure 2.2f) est un périphérique pour ordinateur standard en général dépourvu d'écran et principalement utilisé pour

(bien que les *trackpads* sont généralement utilisé avec un contrôle du curseur relatif). Une tablette tactile retourne la position d'un ou plusieurs points de contact avec la surface. Les *joysticks* sont également des périphériques absolus : ils retournent un vecteur à deux dimensions représentant le degré de rotation du levier par rapport à sa position d'origine. En théorie, les périphériques absolus peuvent permettre un contrôle du curseur absolu ou relatif (voir la Partie 2.2.1 page 23).

2.1.2 Technologies tactiles

La grande majorité des tablettes tactiles reposent sur une dalle capacitive. Cependant, il existe de nombreuses technologies différentes pour percevoir le toucher (voir la Figure 2.3). Dans cette partie, nous décrirons les plus courantes [61, 77, 80, 111, 148, 170, 212].



(a) Technologies capacitives

(b) Technologies résistives

(c) *Diffused Surface Illumination*

(d) *Rear Diffused Illumination*

Figure 2.3 – Exemples de technologies de surfaces tactiles (images en provenance de Gilliot [80])

le dessin, souvent à l'aide d'un stylet.

Technologies capacitives projetées

Les technologies capacitives (voir Figure 2.3a) projetées sont constituées d'une couche conductrice qui recouvre le dispositif. Le contact avec un autre conducteur, en particulier le corps humain, résulte en un changement de capacité qui peut être mesuré. Ces technologies sont utilisées sur la plupart des tablettes tactiles d'aujourd'hui.

Les technologies capacitives projetées sont précises, *multitouch*, disposent d'un faible temps de réponse et sont résistantes. Toutefois, elles sont en général uniquement prévues pour la détection de la peau. Les caractéristiques de l'épiderme peuvent néanmoins être mimées notamment à l'aide de tags capacitifs [56, 121, 176] (certains nécessitent néanmoins un contact cutané pour fonctionner [56, 176]) ou en équipant un stylet avec un embout spécial [2]. La précision de ces stylets est cependant faible en comparaison des stylets à induction.

Les *APIs* fournies par les constructeurs fournissent pas ou peu d'information sur la forme des contacts. Il est cependant parfois possible d'obtenir une approximation de la taille de la surface de contact ou des longueurs du petit et du grand axe de l'ellipse.

Technologies résistives

Les systèmes résistifs (voir Figure 2.3b) sont constitués de deux couches conductrices électrifiées séparées par une cale d'espacement microscopique. Lorsque l'utilisateur touche la surface, la pression exercée crée un contact entre les deux couches. La variation du champ électrique au sein des deux couches permet de déterminer les coordonnées du point de contact.

Les technologies résistives consomment peu d'énergie, sont peu chères et permettent la détection du contact avec n'importe quel objet. Toutefois, elles sont peu précises et ne peuvent souvent détecter la présence que d'un seul contact. Elles ne retournent aucune information sur la forme de ce contact.

Les technologies résistives sont de moins en moins utilisées. On les trouve cependant encore sur des pavés tactiles, des caméras et des boîtiers de navigation GPS, etc.

Technologies à induction

Les surfaces tactiles à induction sont sensibles au champ magnétique d'un périphérique associé, en général un stylet.

Elles sont capables de détecter avec précision la position de cet objet jusqu'à quelques millimètres avant que celui-ci n'entre en contact avec la surface. Souvent, la pression est également disponible grâce à un capteur intégré à la mine du stylet.

2. http://www.fiftythree.com/pencil par exemple.

Les technologies tactiles ne sont cependant pas capables de détecter un autre type contact et ne donnent pas d'information sur la forme du contact.

Il s'agit du type de technologies utilisé par la majorité des tablettes graphiques (Figure 2.2f, à ne pas confondre avec les tablettes tactiles). On en trouve également avec certains ordinateurs portables équipés d'un écran tactile et d'un stylet.

Technologies à flux vidéo

Les technologies à flux vidéo utilisent des caméras. Il en existe plusieurs types. Les FTIR (*Frustrated Total Internal Reflection*), par exemple, s'appuient sur des diodes électroluminescentes (LED) placées sur la tranche d'une plaque en acrylique dépolie et qui illuminent les surfaces de contact avec la plaque. Les DSI (*Diffused Surface Illumination*, voir la Figure 2.3c) utilisent le même principe, mais illuminent également les objets au-dessus de la surface grâce à une surface diffuseuse. D'autres exemples de technologies à flux vidéo classiques incluent celles reposant sur des projecteurs infrarouges placés devant la surface (*Front Diffused Illumination*) ou derrière la surface (*Rear Diffused Illumination*, voir la Figure 2.3d). Dans le premier cas, les objets en contact cachent la lumière infrarouge alors que dans le second, ils la réfléchissent.

Ce type de dispositifs peut parfois détecther un grand nombre d'objets en contact quelle qu'en soit la nature. En outre, une image assez précise des zones de contact peut être obtenue. En fonction du système optique, la précision peut toutefois être moindre et sujette au bruit. De plus, un volume important est souvent nécessaire pour que le champ de vision des caméras englobe l'ensemble de la surface. Cela en fait des dispositifs encombrants.

Les tables tactiles, pour lesquelles l'encombrement est moins un problème, s'appuient souvent sur des technologies à flux vidéo.

Assez récemment, Hodges et al. [104] ont proposé la technologie ThinSight qui miniaturise le système optique au point de pouvoir être intégrée au sein d'un écran LCD. Cela réduit donc considérablement l'encombrement du dispositif. ThinSight commence à être introduit dans les produits Microsoft (par exemple la table Surface) et on peut imaginer qu'elle arrive un jour sur tablette.

Technologies à lasers infrarouges

Au lieu d'un revêtement, la surface tactile est entourée d'un cadre équipé de sources de lumière infrarouge d'un côté et de détecteurs de l'autre. Lorsqu'un objet touche l'écran, il coupe le faisceau lumineux et donc interrompt le signal capté par les détecteurs de lumière.

Ces technologies peuvent être déployées sur des surfaces de très grande taille et sont capables de détecter la position de plusieurs points de contact, de n'importe quel provenance avec une certaine précision. Peu d'information sur la forme de ces

contacts peut toutefois être obtenue : en générale seule leur taille est disponible. Les technologies infrarouges nécessitent également d'entourer la surface par un cadre relativement volumineux.

Ces technologies peuvent typiquemet servir à équiper des murs d'écrans.

Technologies acoustiques

Les technologies acoustiques utilisent des microphones ou des cellules piézoélectriques afin de localiser les sons produits par un contact avec la surface. Parfois, un émetteur ultrasonique est également déployé. Dans ce cas, les microphones détectent la perturbation de l'émission ultrasonique provoquée par un contact avec la surface.

Le principal intérêt des technologies acoustiques est qu'il est possible de rendre tactile pratiquement n'importe quelle surface. Toutefois, elles demandent une calibration précise et sont très sensibles à toute altération (comme les rayures ou même parfois la poussière). Elles sont capables de détecter un nombre limité de points de contact, sont assez peu précises, et ne donnent pas d'information sur la forme des contacts.

2.1.3 Propriétés des dispositifs d'entrée tactiles

Chacune de ces différentes technologies permet un traitement plus ou moins élaboré d'un certain nombre de paramètres. Le tableau 2.1 indique ceux propres aux tablettes.

Nature des objets pouvant être détectés

Certains dispositifs, tels que les tables tactiles s'appuyant sur un flux vidéo, peuvent capter n'importe quel contact quelle que soit la nature de sa source [230]. D'autres au contraire ne peuvent détecter le contact que d'un seul type d'objet. Par exemple, les tablettes graphiques ne sont sensibles en général qu'au seul contact des objets qui leurs sont associés (notamment un stylet). La plupart des tablettes tactiles sont essentiellement prévus pour le contact cutané même s'il est possible de le simuler en utilisant des tags capacitifs [56, 121, 176] ou des stylets spéciaux (voir la Partie 2.1.2).

Nombre de contacts

Le nombre de contacts pouvant être reconnus est variable selon les technologies tactiles (voir la Partie 2.1.2). Un dispositif d'entrée tactile capable de détecter plus d'un point de contact est appelé dispositif d'entrée *multitouch*. Les tablettes tactiles sont en général capables de détecter une dizaine de contacts ce qui est suffisant pour un utilisateur unique.

Forme et propriété de l'objet en contact

Certains dispositifs d'entrée tactiles peuvent informer sur la forme de la zone en contact. Ceci peut permettre d'inférer l'orientation des doigts [225], de faire la différence entre une main posée à plat ou sur la tranche [76, 228, 238] ou d'identifier un objet [28] et certaines de ses propriétés [227], voire même un utilisateur en particulier [66, 106] comme dans Fiberio où les empreintes digitales des utilisateurs sont identifiées lorsqu'ils touchent la surface [106]. Aujourd'hui, les tablettes tactiles capables de retourner davantage d'informations que le diamètre de la zone de contact sont encore rares. Les tablettes Apple, par exemple, ne donnent officiellement accès à aucune information autre que les coordonnées du point de contact [3].

Pression

La pression ou force normale est la composante exercée par le doigt orthogonalement à la surface. Plusieurs travaux ont démontré l'utilité de la pression comme entrée utilisateur avec un périphérique mobile, notamment lorsqu'un retour utilisateur visuel adapté est mis en place [33, 48, 97, 153, 173, 205, 206, 232]. Toutefois, selon Herot et Weinzapfel [97], spécifier une valeur de pression donnée est long (de ∼1,5s à plus de 2s sans retour) et un utilisateur peut difficilement différencier plus de 5 à 7 valeurs [153, 173].

La plupart des tablettes graphiques fournissent cette information. Par contre, durant nos travaux et jusqu'au début de la rédaction de ce manuscrit, à notre connaissance, aucune tablette tactile n'était équipée d'un capteur de pression. Cependant, dernièrement, Apple a commencé à en intégrer dans ses produits [4].

En l'absence de capteurs dédiés, il est possible de simuler la mesure de la pression [9, 13, 33]. On parle alors de pseudo-pression. Deux techniques différentes sont utilisées. La première fait l'hypothèse qu'exercer davantage de force prend plus de temps [9]. Toutefois, en pratique, temps et pression sont faiblement corrélés [9]. La seconde s'appuie sur la forme de la surface du contact [9, 33]. En effet, plus l'on exerce de force, plus le doigt s'écrase sur la surface et donc plus importante est la surface de contact. Cette méthode dépend toutefois de la taille des doigts de l'utilisateur ainsi que de l'orientation du doigt [9, 225]. Arif et Stuerzlinger ont

3. Sur un iPad de Apple, le diamètre du contact n'est pas officiellement accessible, mais peut être tout de même parfois obtenu à travers certaines bibliothèques privées. Ces bibliothèques privées sont des briques logicielles internes au système d'exploitation et qui ne sont pas prévues pour être utilisées directement par les développeurs d'applications. L'information n'est donc pas stable et est sujette à disparition ou modification d'une version à une autre. En fait, Apple détecte automatiquement les applications y faisant appel et bloque leur distribution sur son *store d'applications*.

4. Le *trackpad* du MacBook de Apple sorti en 2015 (le dernier au moment de la rédaction de ce mémoire) intègre des capteurs de pression. C'est également le cas du premier modèle de la montre de la marque. Il est donc probable que les autres dispositifs tactiles de la marque leur emboîtent rapidement le pas. En particulier les tablettes tactiles.

Tableau 2.1 – Propriétés tactiles courantes des tablettes tactiles

Type de technologie :	Capacitive projetée.
Nature du contact :	Doigts uniquement. Parfois également un stylet associé.
Nombre de contacts :	∼10 en général. Parfois seulement 5.
Forme du contact :	Souvent non disponible. Parfois le diamètre de la zone de contact.
Pression :	Non disponible. La pseudo-pression peut permettre de l'estimer dans une moindre mesure.
Force tangentielle :	Non disponible.

présenté une méthode hybride combinant les deux approches afin de minimiser le taux d'erreur [9].

Force tangentielle

La force tangentielle est la composante exercée par le doigt dans le plan de l'écran. Sa mesure est pour l'instant réservée à quelques prototypes académiques [136].

2.1.4 Fusion de l'entrée et de la sortie et pointage sans intermédiaire

Sur une tablette tactile, la surface tactile et le dispositif d'affichage (l'écran) sont colocalisés (ce qui les différencie d'un *trackpad*). On parle d'écran tactile. L'affichage du curseur pour représenter le contact n'est alors pas nécessaire. On parle parfois de système d'entrée direct pour désigner un système d'entrée fusionné avec le dispositif d'affichage. Nous verrons plus bas que si cette propriété permet d'améliorer la directivité de l'interaction, il y a un prix, celui de l'occultation visuelle et de la perte de précision.

Directivité

Le terme *manipulation directe* à été introduit par Shneiderman [201] pour désigner un système ayant les propriétés suivantes :

— Représentation continue des objets d'intérêt,

— Actions physiques plutôt que syntaxe complexe,

— Opérations rapides, incrémentales, réversibles et dont les effets sur l'objet d'intérêt sont visibles immédiatement.

Bien que les interactions directes ne doivent pas être considérées comme nécessairement et universellement meilleures [109, 120], un système qui les met en œuvre est généralement jugé :

1. moins demandeur en ressources cognitives,
2. plus agréable à utiliser,
3. nécessitant peu de messages d'erreur,
4. plus facile à apprendre et à maîtriser,
5. il est possible de voir immédiatement le résultat d'une action et de l'ajuster si besoin [109, 201].

On considère en effet que les interactions directes permettent de réduire ce que Norman appelle « le gouffre d'évaluation » – qui représente l'écart entre l'interprétation par l'utilisateur des sorties du système et leur signification réelle – et « le gouffre d'exécution » – qui représente l'écart entre l'action de l'utilisateur sur le système et l'action qu'il aurait fallu faire pour atteindre son but [109, 162].

Hutchins et al. soutiennent que la directivité d'une interface n'est pas une métrique objective, mais le résultat d'une appréciation subjective, un sentiment concernant l'interaction avec un système [109]. Il ne s'agit donc pas d'une dichotomie, mais d'un spectre continu : une interface peut être jugée plus directe qu'une autre.

Une des propriétés importantes favorisant la sensation de directivité sur une interface est l'utilisation des éléments en sortie du système (par exemple la représentation graphique d'un objet) comme sources d'entrées potentielles. Un utilisateur a alors l'impression de manipuler directement l'objet que le système représente. Sur un ordinateur standard, le sentiment de directivité peut être amoindri par l'utilisation d'un système d'entrée indirecte (en général une souris) et le recours nécessaire à l'affichage d'un curseur. Du point de vue de l'utilisateur, on agit sur la souris, qui elle même agit sur le curseur, qui lui-même, finalement, agit sur les objets d'intérêt. Avec un système d'entrée directe, il devient possible de toucher avec le doigt l'objet d'intérêt. On agit directement sur celui-ci, sans intermédiaire, ce qui favorise le sentiment de directivité [202, 235].

Beaudouin-Lafon a proposé le modèle de l'interaction instrumentale [29] qui est centré sur les objets d'intérêt. Dans ce modèle, on interagit sur les objets à travers une séquence de médiateurs appellés *instruments* : Barre de défilement, Menu, mais aussi souris et autres périphériques d'entrée sont des instruments pouvant agir sur l'objet d'intérêt, mais aussi les uns sur les autres [29, 30]. Par exemple, si l'objet d'intérêt est un document, l'utilisateur peut agir sur l'instrument souris qui agit sur l'instrument barre de défilement qui elle même fera défiler l'objet d'intérêt document (voir la Figure 2.4). L'interaction instrumentale introduit trois métriques fondamentales pour une interface :

Le degré d'indirection mesure la distance spatiale et temporelle entre l'instrument et sa cible.

Le degré d'intégration mesure le ratio entre les degrés de liberté en entrée et en sortie de l'instrument. Par exemple une barre de défilement (1D) manipulée par une souris (2D) a un degré d'intégration de 1/2.

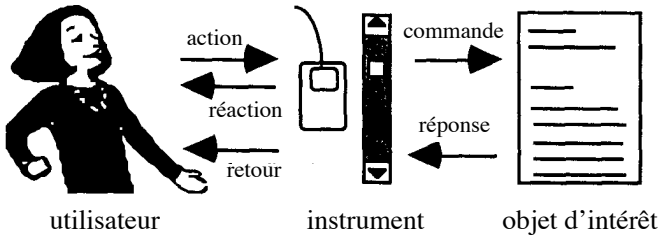


Figure 2.4 – Interaction Instrumentale [29, 30]

Le degré de compatibilité mesure la similarité entre les actions sur l'instrument et sa réponse. Par exemple, le défilement d'un document traditionnel avec la molette d'une souris a un faible degré de compatibilité, car faire tourner la molette vers le bas fait défiler le document vers le haut et inversement. Il est d'ailleurs intéressant de noter que Apple a inversé le défilement depuis OS X 10.6 ce qui propose donc un meilleur degré de compatibilité vis-à-vis de la méthode traditionnelle : faire tourner la molette vers le bas fait défiler le document vers le bas et inversement. Pour Apple, la raison principale de ce choix est sans doute toutefois une meilleure compatibilité par rapport à l'interaction sur tablette (qui elle ne peut pas être inversée).

Occultation

Fusionner la surface d'affichage avec une surface tactile implique inévitablement que les doigts et la main cachent une partie de l'écran lors de l'interaction [202, 221]. Pour cette raison, alors que sur un ordinateur standard, les menus contextuels s'ouvrent en général en dessous et à droite du curseur, sur une tablette tactile ils s'ouvrent en général au-dessus du doigt sous peine d'être occultés.

Précision

Fusionner écran et dispositif de pointage a également un impact sur la précision du pointage (voir la Partie 2.2) lorsque les doigts sont utilisés [171, 220]. En effet, en plus des problèmes d'occultation, la surface du doigt en contact est élevée. Il est donc difficile d'estimer la position exacte que souhaite atteindre un utilisateur. D'après Holz et Baudisch, celle-ci dépend de l'inclinaison du doigt, de sa rotation sur l'axe normal, mais aussi de chaque utilisateur et de leur modèle mental propre [107]. On parle du problème du gros doigt (*fat finger problem*, voir la figure 2.5). Concernant les tablettes commerciales il est probable que le barycentre de la zone de contact soit prit pour référence (nous n'avons pu trouver d'information précise à ce propos). Dans la littérature, certains travaux suggèrent de décaler cette position vers le haut de la zone de contact ce qui correspondrait mieux à l'idée que se font les utilisateur

Figure 2.5 – Le problème du gros doigt

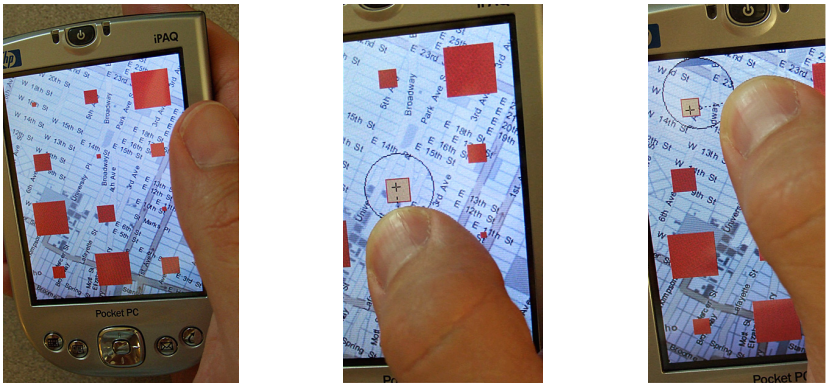


Figure 2.6 – Shift permet de pointer avec précision en révélant les cibles masquées par le doigt [220]

de la position du point de contact [33, 220]. Le *Generalized Perceived Input Point Model* de Holz et Baudisch peut permettre de réduire considérablement la taille minimale d'un bouton, mais nécessite plus d'information que les surfaces tactiles ne peuvent habituellement en délivrer (orientation précise des doigts ou empreintes digitales) [107]. Lorsque le besoin en précision est importante sur une surface tactile traditionnelle, des techniques telles que *Shift* [220] (voir la figure 2.6) ou TapTap [182] sont nécessaires.

2.1.5 Retour utilisateur

Bien que la fusion avec l'écran permette un retour visuel, une tablette tactile ne donne pas de retour haptique avec un clavier ou des boutons physiques. Un écran tactile étant une surface de verre homogène, l'utilisateur ne peut rien percevoir de

plus que la dichotomie du contact et de l'absence de contact [102]. Il est cependant possible de mettre en place des retours vibrotactiles ou audio informant, par exemple, lorsqu'un bouton a été pressé avec succès. D'autres technologies comme *SurfPad* [54] ou *TeslaTouch* [26] permettent de modifier la friction lors du déplacement sur la surface grâce à des micro-vibrations.

2.2 Pointage sur tablettes multi-tactiles

Le pointage occupe une place capitale sur ordinateurs traditionnels. L'interaction repose en effet majoritairement le paradigme *WIMP* : « *Windows*, *Icons*, *Menus* et *Pointing device* ». Celui-ci s'appuie exclusivement sur le pointage : ouvrir un menu, cliquer sur un bouton, une icône, etc. Seules quelques rares commandes comme l'utilisation de la molette de la souris pour faire défiler un document font exceptions. Sur tablette tactile, l'absence de clavier physique et donc l'indisponibilité des raccourcis clavier accroît encore davantage le nombre de tâches de pointage sur l'écran.

La majorité des tablettes tactiles commercialisées aujourd'hui sont essentiellement prévues pour le contact cutané et ne sont pas conçues pour être utilisé avec un stylet (technologie capacitive projetée, voir la Partie 2.1.2). Nous nous intéresserons essentiellement à l'interaction avec les doigts.

On définit le pointage comme la sélection des coordonnées d'un (voire plusieurs) point d'interaction au sein d'un espace. Nous appellons ces points d'interactions « pointeur ». Sur un ordinateur traditionnel, le pointeur à presque toujours une position définie et est généralement représenté par un curseur à l'écran (il est indispensable pour le pointage). C'est inutile sur tablette puisque la surface tactile est superposée à l'écran et que son espace de coordonnées est mis en correspondance directe avec les pixels (voir la Partie 2.1.4). La position des doigts de l'utilisateur indique donc de manière évidente (mais imprécise) la position du ou des pointeurs.

Dans cette partie nous nous intéresserons dans un premier temps aux différents types de contrôle du pointeur (absolu/relatif, en position/en vitesse). Nous parlerons ensuite des différents états de pointage et du modèle de Buxton [44]. Nous terminerons sur l'estimation des performances et sur l'apprentissage avec les techniques reposant sur le pointage.

2.2.1 Contrôle du pointeur

Les coordonnées du pointeur sont transformées à partir des coordonnées d'un périphérique de pointage par le biais d'une fonction de transfert. Une fonction de transfert est un modèle mathématique de la relation entre l'entrée et la sortie d'un système. Dans notre cas, l'entrée du système correspond à l'information envoyée par le dispositif d'entrée, et la sortie, la position du ou des pointeurs.

Il existe différentes classes de fonction de transfert, et donc différents types de pointage. L'interaction classique avec une tablette tactile repose exclusivement sur un contrôle de position (ou contrôle d'ordre zéro), et cette position est définie de manière absolue (par opposition par exemple au cas de la souris, qui ne délivre que des informations de déplacements).

Bien que ce soit lié, il ne faut pas confondre dispositif d'entrée absolu et contrôle du curseur absolu ou dispositif d'entrée relatif et contrôle du curseur relatif (voir la Partie 2.1.1). Un dispositif d'entrée relatif permet uniquement un contrôle relatif. Un dispositif d'entrée absolu, toutefois, permet à la fois un contrôle relatif et un contrôle absolu. Par exemple, un *trackpad* est un dispositif d'entrée absolu qui contrôle généralement le pointeur de manière relative.

Contrôle en position absolu

Lors du contrôle en position absolu, la fonction de transfert établit une relation bijective entre les coordonnées transmises par le périphériques et les coordonnées sur l'espace de pointage.

Par exemple, dans le cas d'une tablette graphique utilisée avec un ordinateur standard, toucher avec le stylet le coin supérieur gauche de la tablette téléporte instantanément le curseur sur le coin supérieur gauche de l'espace de pointage.

Le pointage absolu est celui généralement utilisé avec une tablette graphique ou une tablette tactile. La fonction de transfert fait coïncider la position du pointeur avec les coordonnés d'un des pixels de l'écran se trouvant sous la zone de contact.

Un pointage absolu n'est pas possible avec un périphérique relatif (voir la Partie 2.1.1).

Contrôle en position relatif

Le pointage en position relatif est celui des souris, des *trackballs* et des *trackpads*. C'est le type de pointage le plus commun. La fonction de transfert calcule le déplacement du pointeur en fonction du déplacement du périphérique d'entrée [53, 80]. En général, elle n'est pas linéaire afin d'optimiser le temps de déplacement du pointeur et la précision du pointage [53].

Contrôle en vitesse

Le pointage relatif est celui des *trackpoint* [188]. Il est assez peu répandu. La fonction de transfert calcule non pas le déplacement du pointeur, mais sa vitesse de déplacement en fonction des valeurs du périphérique d'entrée (par exemple la force exercée sur un *trackpoint*).

2.2.2 États de pointage

L'entrée d'une souris peut être modélisée en deux états (Figure 2.7a) : un état où les mouvements de la souris sont interprété comme le déplacement du curseur, et un état, bouton enfoncé, où les mouvements de la souris sont interprétés dans un mode alternatif en général fonction de sa position [44]. Buxton appelle ce mode *drag* ou glissé (pour être exact, il s'agit en fait d'un quasimode, voir le Chapitre 3).

L'état de déplacement de la souris peut servir à afficher des infobulles en survolant les objets graphiques d'une scène. Le *drag* peut servir à définir une zone de sélection. Ces deux états sont actifs car les actions de l'utilisateur sur le périphérique sont capturées par le système.

Au modèle de Buxton, on peut rajouter un état passif lorsque la souris est soulevée de son support. Ceci est souvent utilisé pour rallonger la course du curseur et atteindre une cible autrement hors de porté. On parle de débrayage ou *clutching* [114, 158]. Un état passif est moins utile qu'un actif puisque les actions ne peuvent pas être interprétées par le système.

Si la souris est composée de plusieurs boutons, chaque combinaison de boutons possibles donne accès à un état actif différent.

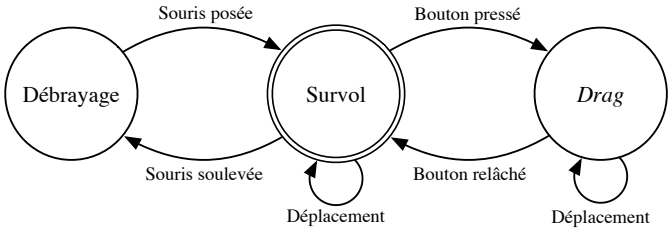
Les *trackpads* sont souvent associés à un ou deux boutons mais sur tablette tactile, le pointage se fait sans bouton. Le modèle de Buxton ne comporte donc que deux états (Figure 2.7b) [44], dont un seul actif : lors que le doigt est en contact avec la surface, et lorsqu'il n'y a aucun contact.

Le *multitouch* permet également d'ajouter des états actifs supplémentaires, par exemple en fonction du nombre de doigts, des postures, etc. Cependant, pour le pointage, la précision est alors moindre à cause de l'occultation engendrée 2.1.4). De plus, la présence de plusieurs points de contact entraînent une ambiguïté sur celui devant être utilisé comme référence. En pratique, ces états ne sont pas utilisés pour pointer sur tablette tactile.

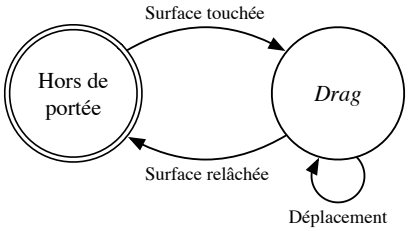
2.2.3 Performances

Les performances du pointage sur surface tactile peuvent être modélisées puis prédites à l'aide de la loi de Fitts [70, 87, 142, 155]. Plusieurs études ont comparé les performances d'un écran tactile avec le doigt avec une souris et parfois un stylet [58, 73, 191, 197]. Leurs résultats montrent que, pour les cibles suffisamment grandes, les performances de pointage en termes de précision et de vitesse des écrans tactiles sont supérieures à celle du stylet et de la souris. Toutefois, lorsque la cible devient trop petite – 5mm dans l'expérience de Cockburn et al. [58] et 8 pixels dans l'expérience de Sasangohar et al. [191] – la précision du pointage au doigt s'effondre brusquement. Cela peut s'expliquer par le « problème du gros doigt » (voir la Partie 2.1.4).

Lee et Zhai ont également montré que l'utilisation de retour utilisateur audio



(a) États de pointage d'une souris à un seul bouton



(b) États de pointage d'une surface tactile

Figure 2.7 – Machine à états du pointage selon le modèle de Buxton [44]

ou vibrotactile pouvait permettre d'améliorer la rapidité et la précision du pointage avec le doigt [138].

Toutefois, dans le cas de la sélection de commandes le pointage ne peut être considéré seul. La petite surface d'affichage et la nécessité d'utiliser des boutons suffisamment grands (voir la Partie 2.1.4) forcent à utiliser des techniques de défilement de liste et/ou une organisation hiérarchique profonde des menus : les menus contiennent des sous-menus qui contiennent d'autres sous-menus et ainsi de suite. Ainsi la sélection d'une commande, par exemple, est souvent associé avec une ou plusieurs interactions intermédiaires avant de pouvoir finalement pointer sur l'objet graphique représentant cette commande. Ceci dégrade sensiblement les performances.

2.2.4 Exploration et apprentissage

Les techniques d'interaction reposant sur le pointage sont très adaptées aux utilisateurs novices. Elles demandent en effet peu ou pas d'apprentissage car elles s'appuient sur la reconnaissance de cibles graphiques affichées à l'écran plutôt que la mémorisation d'actions à effectuer [162]. Un utilisateur novice ne connaît pas encore la manière pour interagir avec le système, et donc repose fortement sur la recherche visuelle [59]. Norman parle de connaissances « dans le monde », qui n'ont donc pas besoin d'être mémorisées, plutôt que « dans la tête » [159].

Toutefois, les tâches de pointage permettent peu d'évolution de performance car

le temps de pointage est restreint par des caractéristiques motrices. Pour cette raison, sur ordinateur standard, des raccourcis claviers sont souvent disponibles pour les commandes les plus fréquentes ce qui permet de minimiser grandement le temps de sélection [51, 85]. Toutefois l'absence de clavier physique sur tablette tactile n'en permet pas l'usage et les utilisateurs experts se retrouvent « piégés en mode débutant » [59].

2.3 Gestes

Les gestes apportent une alternative puissante au pointage. En fait, dans une certaine mesure, le pointage peut être considéré comme la forme la plus simple d'un geste : un simple contact entre un doigt et la surface, ce qui est interprété comme une coordonnée X et Y par le système. Prendre en compte l'ensemble du geste, mais aussi éventuellement sa cinématique, est évidemment plus expressif. Cela se fait toutefois au prix d'un temps d'exécution inévitablement plus important, mais aussi d'une complexité accrue (à la fois pour l'utilisateur et le système) et donc d'un apprentissage plus difficile et plus long (voir les Partie 2.3.3 et 2.3.4).

La terminologie associée aux gestes n'est pas cohérente et est souvent ambigüe dans la littérature en IHM. Le terme « geste » caractérise l'ensemble des mouvements de la main et du corps [46, 242]. Or, ce terme est souvent utilisé pour désigner le déplacement des points de contacts sur une surface tactile [6, 239, 240], c'est à dire une suite de coordonnées éventuellement horodatées [5]. Buxton appelle cela une « *mark* » (que l'on peut traduire par « marque ») et insiste sur le fait qu'il ne s'agit que du résultat du geste et non le geste en lui même [46, 126]. Les termes « *hand drawn gestures* » [235], « *hand drawn mark* » [236], « *stroke* » [7] ou encore « *gesture stroke* » [242] ont aussi été utilisés.

Nous commencerons par classifier les différents types de gestes en nous intéressant plus particulièrement à ceux pouvant être détectés sur une surface tactile. Nous verrons ensuite les différents modèles de performances, parlerons des gestes pouvant être effectué de manière instinctive ou avec peu ou pas d'apprentissage puis nous intéresserons aux guides permettant à un utilisateur novice d'apprendre et découvrir les possibilités apportés par une interface gestuelle. Nous terminerons sur la transition de novice à expert puis les freins à l'adoption des gestes.

2.3.1 Taxonomie des gestes

Les gestes sont versatiles et hautement variés : pratiquement tout ce que l'on peut faire avec les mains (ou même le corps) est susceptible d'être interprété [233]. Ainsi, il est difficile de capturer et catégoriser la diversité des gestes. Plusieurs

5. L'horodatage (en anglais *timestamping*) est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique.

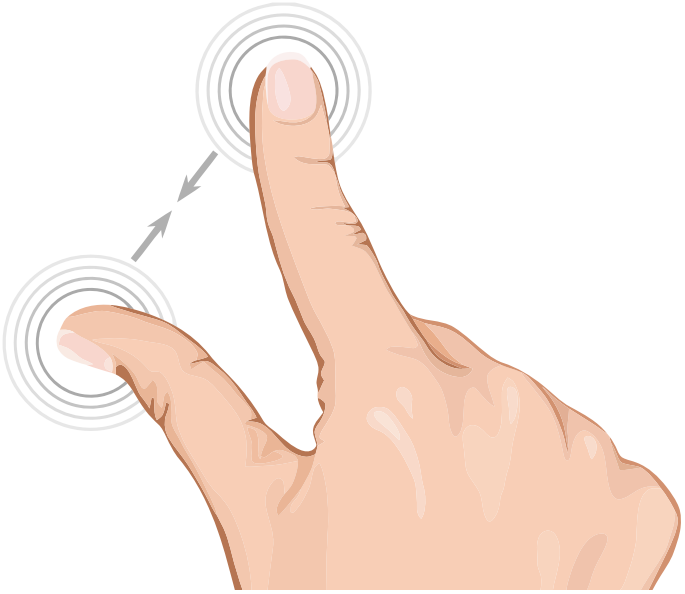


Figure 2.8 – Geste *pinch*. Il permet la modification de l'échelle ou le zoom, un effet qui peut être combiné avec une translation et une rotation.

taxonomies ont été proposées [11, 119, 233, 242]. Celle de Wobbrock et al. [233], qui s'intéresse en particulier aux gestes en surface et repose sur l'observation du comportement des utilisateurs, est souvent considérée comme une référence. Elle se compose de quatre axes : forme, nature, lien (*binding*) et flux (voir le tableau 2.2).

Forme

La forme d'un geste est la manière dont il est effectué. Wobbrock et al. distinguent :

posture statique quand la main est maintenue à une position donnée,

posture dynamique quand la main reste à une position donnée, mais change de posture,

posture statique avec trajectoire quand la main change de position sur la surface sans modifier sa posture,

posture dynamique avec trajectoire quand la pose de la main change en même temps qu'elle se déplace sur la surface.

Wobbrock et al. distinguent également les cas particuliers où il y a un seul point de contact avec la surface (contact unique fixe et contact unique avec trajectoire).

Zhai et al. appellent le contact unique fixe « geste d'ordre 0 ». Les gestes d'ordre 1 sont les gestes uni-trait (*unistroke*) et à un seul doigt. Au-delà, les gestes à plusieurs

Tableau 2.2 – Taxonomie des gestes de Wobbrock et al. [233]

Forme	Posture statique
	Posture dynamique
	Posture statique avec trajectoire
	Posture dynamique avec trajectoire
Nature	Symbolique
	Physique
	Métaphorique
	Abstraite
Interdépendance	Dépendant de l'objet
	Dépendant de l'environnement
	Indépendant
	Mixte
Flux	Discret
	Continu

doigts et/ou composés de plusieurs traits ne sont pas différenciés et sont appelés gestes d'ordre supérieur [242].

Nature

Wobbrock et al. distinguent quatre types de nature différents pour un geste :

Les gestes symboliques qui correspondent au dessin d'un symbole. Par exemple les caractères alphabétiques.

Les gestes physiques qui sont ceux mimant exactement les effets physiques, c'est-à-dire le type d'interaction que l'on a avec des objets réels. Par exemple un *scroll* sur un document mime la manière dont on peut faire glisser le papier avec un doigt.

Les gestes métaphoriques qui calquent un comportement réel et le réifient dans un autre contexte ou d'une autre manière, différente de celle que l'on pourrait observer dans le monde réel. Par exemple utiliser deux doigts pour « marcher » sur l'écran ou poser la tranche de sa main comme un « mur » virtuel et marquer une information comme privée.

Les gestes abstraits qui ne font appellent à aucune connaissance ou convention et doivent être appris à partir de zéro.

Similairement à la nature d'un geste au sens de Wobbrock et al., Zhai et al. [242] distinguent les gestes **analogues** et **abstraits**. Les gestes analogues sont ceux qui miment les effets physiques ou conventionnels du monde réel. Par exemple, un geste

de déroulement (*pan*) peut être utilisé pour faire défiler une page WEB de la même manière qu'on déroulerait un papier physique. Le *pinch* (geste d'agrandissement ou de rétrécissement avec la pince pouce-index, voir Figure 2.8) est un autre exemple de geste analogue. Au contraire, un geste abstrait est arbitraire (les gestes abstrait au sens de Zhai et Kristensson regroupent les gestes symboliques et les gestes abstraits au sens de Wobbrock et al.

Interdépendance

L'interdépendance (*link* dans la taxonomie de Wobbrock et al. [233]) détermine quel type d'information sur le système a besoin un utilisateur avant d'effectuer un geste.

Les gestes dépendant de l'objet nécessitent de connaître certaines propriété de l'objet d'intérêt (typiquement sa position). Par exemple, presser un bouton nécessite de savoir où il se trouve à l'écran.

Les gestes dépendant de l'environnement demandent de l'information sur le monde. Par exemple glisser un objet en dehors de l'écran nécessite de savoir où sont les bords de l'écran.

Les gestes indépendants, au contraire, ne requièrent aucune information et peuvent être effectués n'importe où sur la surface.

Les gestes à dépendances mixtes sont des gestes indépendants sur certains aspects, mais dépendants sur un autre. Par exemple un geste à deux mains où une main doit agir sur un objet alors que l'autre est libre d'interagir n'importe où.

En général, plus un geste est dépendant plus il gagne en capacité informative puisque l'on peut alors tirer parti des objets graphiques ou de la position relative plutôt que de se reposer uniquement sur la forme du geste.

Geste discret ou geste continue

Un geste peut être interprété dans son ensemble, ou, au contraire au fur et à mesure. Wobbrock et al. appellent cette propriété « *flux* ».

Les gestes discrets sont des gestes interprétés en une seule fois, après avoir été exécutés entièrement.

Les gestes continus sont des gestes dont l'interprétation a lieu incrémentalement tout au long de leurs exécutions. Par exemple gommer avec le doigt un dessin ou redimensionner une image avec un geste d'écartement (Figure 2.8).

2.3.2 Temps d'exécution d'un geste

La vitesse d'exécution d'un geste dépend bien entendu du degré de familiarité avec ce geste.

Plusieurs modèles tentant de prédire le temps d'exécution d'un geste sur une surface ont été proposés dans la littérature. Entre autres, ces modèles donnent une bonne indication des paramètres à prendre en compte.

La loi de Viviani par exemple s'appuie sur la courbure du mouvement [218, 219]. Elle stipule que plus la courbure d'un geste sur une surface est importante plus la vitesse est faible. Le modèle de Isokoski [112] utilise le nombre de lignes droites comme indicateur du temps de production. Il fait l'hypothèse que dessiner une ligne droite demande le même temps, quelle que soit la longueur du segment. Ce modèle dispose d'une précision assez faible, mais permet d'obtenir une prédiction facile et rapide. Enfin, le modèle *CLC* (*Curves, Lines and Corners*), proposé par Cao et Zhai, est parmi les plus précis à ce jour [50]. Il est relativement plus complexe que les autres. Comme son nom l'indique, il s'appuie sur les lignes droites, les angles et les courbes pour prédire le temps d'un geste sur une surface.

Toutefois, l'ensemble de ces travaux s'est concentré sur la prédiction des gestes du premier ordre au sens de Zhai et al. [242], c'est-à-dire les gestes d'un seul tenant et à un seul point de contact. Il n'existe à ce jour et à notre connaissance aucune méthode fiable pour estimer la durée d'un geste d'ordre supérieur — pouvant être composé de plusieurs traits et/ou de plusieurs points de contact en parallèle — autre qu'une étude utilisateur.

De plus, ces travaux permettent uniquement de calculer la vitesse d'exécution du geste par un expert. La vitesse d'exécution d'un geste par un novice dépend de la méthode d'apprentissage mise en place (voir la Partie 2.3.4). La durée et la quantité d'entraînement nécessaire pour atteindre les performances d'un expert avec le geste dépendent de la complexité de ce geste, mais aussi, à nouveau, de la méthode d'apprentissage.

À titre comparatif, Appert et Zhai comparèrent l'utilisation de gestes simples avec des raccourcis clavier [7]. Ils obtinrent en moyenne un temps d'exécution optimal équivalent. Ce temps d'exécution optimal correspond au moment où un utilisateur atteint le maximum de ses performances et inclut temps de remémoration et temps de geste.

2.3.3 Gestes instinctifs

Dans les premières approches pour mettre en oeuvre une interface gestuelle les auteurs ont essayé d'extraire des gestes pouvant être utilisé instinctivement et quasi sans apprentissage. L'objectif est de créer une association entre le geste et la commande suffisamment forte pour que celle-ci se mémorise rapidement et reste en mémoire. Plusieurs études tentèrent d'évaluer la concordance des gestes entre utilisateurs [84, 233, 234, 235]. L'objectif est de déterminer si une méthode d'interaction dite « naturelle[6] » existe, c'est à dire un ensemble d'actions suffisamment

6. Raskin pointa du doigt que les termes « intuitif » ou encore « naturel » sont particulièrement vagues et peu informatifs [174, 175]. Bien qu'il existe dans la littérature des tentatives pour leur

évidentes pour qu'elles soient intuitivement exécutées par un utilisateur, avec peu ou aucun apprentissage.

Les premières études trouvèrent un degré de concordance intra-utilisateur et inter-utilisateur élevé ce qui fut considéré comme fortement en faveur de cette hypothèse [84, 234, 235] . Par exemple, les utilisateurs ont tendance à encercler pour déterminer un cadre ou une sélection, utiliser des flèches pour déplacer un objet ou tracer un X pour le supprimer.

Toutefois, plus récemment, Wobbrock et al. montrèrent que cette concordance se rencontrait essentiellement pour les fonctions dont le geste était similaire à l'interaction avec un objet physique (geste de nature physique) [233]. Les fonctions « déplacer un peu », « déplacer beaucoup » reçurent un fort taux de concordance (score de 1), les fonctions « dupliquer », « tourner », « sélectionner » et « grouper » reçurent un taux de concordance moyen (score de 0.6 à 0.4) et enfin, les fonctions plus abstraites telles que « annuler », « accepter », « demander de l'aide », « rejeter », « suivant » et « couper » reçurent au contraire un taux de concordance faible (moins de 0.2).

Le nombre de fonctionnalités susceptibles d'être associées à un geste sans mécanisme d'apprentissage est donc fortement limité. Un système de guidage et d'apprentissage est nécessaire dans la plupart des cas.

On peut noter que la dite « intuitivité » ou « naturalité » des interfaces gestuelles d'aujourd'hui fut récemment durement critiqué, notamment par Norman [161, 163].

2.3.4 Guides

Plusieurs techniques peuvent être mises en place afin de guider l'utilisateur pour apprendre à effectuer les gestes disponibles. Le principal défi d'un guide est qu'il s'agit d'un artefact. Un guide tend à momentanément capturer l'attention de l'utilisateur, il ne fait pas lui même partie de la tâche. Indispensable pour un utilisateur novice, il doit se faire le plus discret possible et tendre à s'effacer lorsque celui-ci gagne en expertise.

Les guides de gestes peuvent être classifiés selon les deux axes suivants : **directivité** (direct/indirect) et **staticité** (statique/dynamique).

Un guide direct s'affiche à l'endroit même où est effectué le geste de manière, par exemple, à être suivi du doigt. L'avantage est que l'utilisateur n'a pas à détourner son attention de ce qu'il fait. Toutefois, ils peuvent poser des problèmes d'occlusion car ils sont souvent recouverts pendant l'exécution du geste.

donner une définition précise [229], il s'agit généralement de mots passe-partout qu'il est préférable d'éviter.

Un guide indirect est au contraire affiché à l'écart et ne pose donc pas de problème d'occlusion.

Un guide statique est un guide qui, hormis le fait de s'afficher et de se masquer, est indépendent des actions de l'utilisateur.

Un guide dynamique a la capacité d'évaluer la progression du geste et d'y réagir en temps réel.

Cette classification se rapproche de celle de Anderson et Bischof [3] (voir le Tableau 2.3).

Tableau 2.3 – Relation entre directivité et statisticité et les catégories de Anderson et Bischof [3]

	Indirect	**Direct**
Statique	*crib-notes*	*static-tracing*
Dynamique		*dynamic-tracing*

Guides statiques indirects

L'exemple le plus évident de guide statique indirect est l'« anti-sèche » (ou *cheat sheet*). Les anti-sèches affichent sur l'écran un panneau contenant une représentation de l'ensemble des gestes disponibles ainsi que les fonctionnalités qui leur sont associées [132] (voir la Figure 2.9). Appert et Zhai [7] remplacèrent les indices de raccourcis clavier disponibles dans les menus linéaires traditionnels par une représentation des raccourcis gestuels. Bragdon et al. [40] proposèrent une barre d'outils qui, au lieu d'exécuter directement une commande, affiche un tutoriel animé du geste à effectuer. Les gestes peuvent être représentés par leur trace [7, 132], par des pictogrammes [42, 68], des animations [40] ou des vidéos [76] (notamment utilisées sur Mac OS X pour indiquer les gestes disponibles avec un pavé tactile).

Guides statiques directs

Au lieu de s'afficher dans un panneau séparé, les guides statiques directs apparaissent sous le doigt ou le stylet de l'utilisateur [3, 132]. Ceux-ci peuvent être immobiles [3] ou animés [132] (bien que cette animation ne dépende pas de la manière dont l'utilisateur exécute son geste). Kristensson et Zhai ont proposé un autre exemple de guides statiques directs [123] qui étend *ShapeWriter* [241] à la sélection de commandes. *ShapeWriter* est une technique d'entrée de texte gestuelle sur clavier virtuel qui permet d'écrire un mot en reliant successivement, approximativement et sans relever le doigt chacune des lettres qui le composent. L'écriture du mot forme donc un geste défini par la position des lettres. Le guide de ce geste est donc le clavier virtuel.

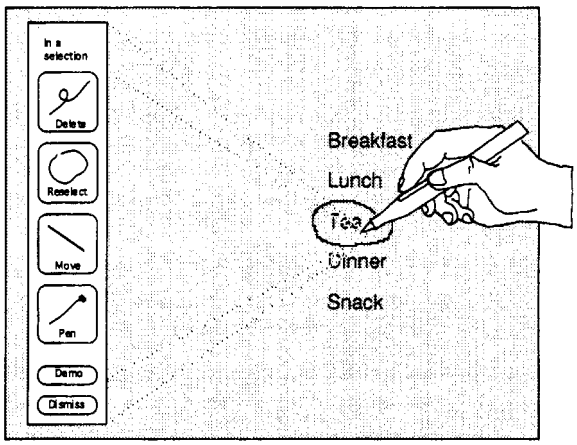


Figure 2.9 – Anti-sèche dans Tivoli [132]

Guides dynamiques directs

Les guides dynamiques s'appuie sur une interprétation continue du geste pour réagir en temps réel à sa progression [25]. Ainsi, OctoPocus [25] affiche sous le curseur l'ensemble des tracés disponibles et se met à jour au fur et à mesure de la progression du mouvement : en fonction du chemin parcouru, les gestes de moins en moins probables s'amincissent petit à petit puis disparaissent tandis que les gestes les plus probables se mettent en avant (voir Figure2.10). Certains guides dynamiques ne permettent pas de découvrir l'ensemble des gestes. Par exemple SimpleFlow ressemble à OctoPocus, mais n'affiche la représentation que d'un seul geste [34]. Kristensson et Denby proposent une approche similaire en affichant un ensemble de points colorés autour du point de contact [122]. La couleur de chaque point indique la probabilité pour le geste d'être reconnu si le mouvement se prolonge dans cette direction. Ghomi et al. s'inspirèrent d'OctoPocus pour guider l'apprentissage de raccourcis basés sur une posture statique de la main [79]. Au sein de cette thèse, nous avons proposé les Lettres Augmentées [185] qui combinent une lettre unistroke avec un *Marking menu* [125, 126, 127, 128, 132, 133] (voir la Partie 5.1).

Guides dynamiques indirects

ShadowGuide annote dynamiquement l'ombre des mains telles qu'elles sont capturées par une table multi-tactile afin d'indiquer les mouvements possibles [76].

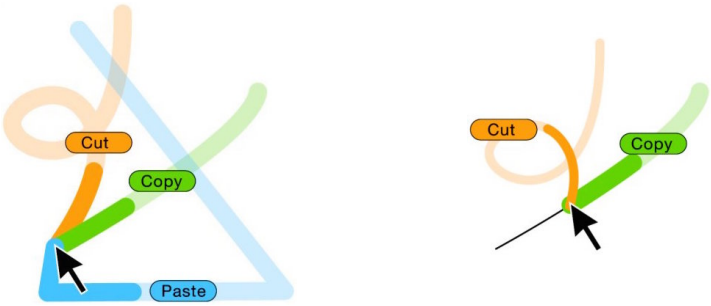


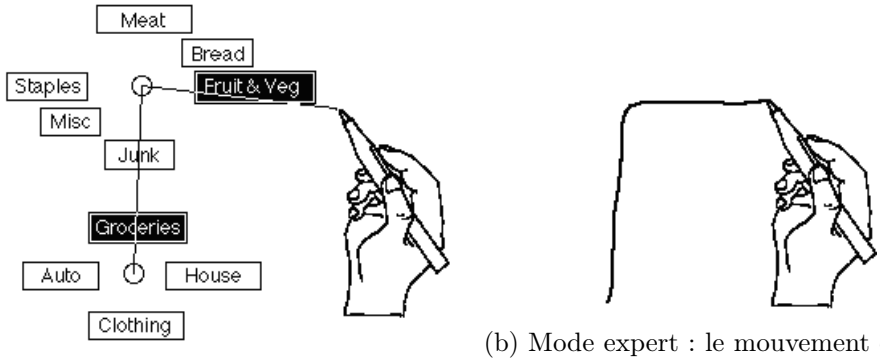
Figure 2.10 – OctoPocus : Guide dynamique qui se met à jour au fur et à mesure du geste [25]

Autres propriétés des guides gestuels

Bau et Mackay proposèrent un espace de classification en identifiant deux types de retour utilisateur le retour sur action (*feedback*) et le retour par anticipation (*feedforward*) [25]. Les retours par anticipation se différencient par (1) leur niveau de détail, et (2) leur fréquence de rafraîchissement. Les retours sur action se différencient par (1) la manière dont ils évaluent les actions de l'utilisateur, (2) le nombre de gestes parmi l'ensemble des gestes disponibles sur lesquels ils donnent de l'information, (3) leur fréquence de rafraîchissement, et (4) le type de représentation utilisé.

Menus Gestuels

Les menus gestuels sont une classe particulière de guides dynamiques qui permettent une exploration sémantique des commandes disponibles [14, 18]. Ils permettent souvent une prévisualisation [75] et une exploration interactive des différentes fonctionnalités au fur et à mesure de l'avancement du geste et donc un apprentissage par la pratique [76, 132]. Le revers de la médaille est qu'ils impliquent souvent des contraintes plus importantes sur la forme des gestes supportés. Le plus célèbre exemple de menu gestuel est le *Marking Menu* [125, 126, 132, 133] (voir la Figure 2.11). Un *Marking Menus* est un menu circulaire s'ouvrant au lieu du point de contact et qui permet de sélectionner un élément en traçant un rayon dans sa direction (Figure 2.11a). Il est possible de mettre en place une hiérarchie en ouvrant un second menu une fois que le premier élément est atteint [127]. Une fois mémorisé, le geste d'un *Marking Menu* peut être effectué sans ouvrir le menu (Figure 2.11b). Il devient donc un raccourci gestuel (en fait l'affichage en mode novice n'est déclenché qu'après un léger délai, voir la Partie 3.2 sur les délimiteurs). De nombreuses variantes des *Marking Menus* ont été proposées dans la littérature [17, 19, 90, 139, 164, 165, 169, 185, 243, 244]. Par exemple, Zhao et Balakrishnan proposèrent d'utiliser plusieurs traits plutôt qu'un seul pour les *Marking Menus*



(a) Mode novice : le menu est affiché

(b) Mode expert : le mouvement du menu devient un raccourci gestuel

Figure 2.11 – Marking Menus [125, 126, 127, 128, 132, 133]

hiérarchiques [244]. Zhao et al. tirèrent parti de la position de départ pour augmenter le nombre de possibilités [243]. Dans le même but, Bailly et al. utilisèrent la courbure du geste [17]. Pook et al. et Guimbretière et Winograd fusionnèrent la sélection de commande avec le contrôle continu [57, 90, 169]. Lepinski et al. combinèrent le *Marking Menu* et la posture multi-tactile [139]. Des *Marking Menu* basés sur l'orientation d'un téléphone furent même mis à l'étude [164]. Les menus gestuels n'impliquent pas obligatoirement une posture dynamique. Par exemple *Finger Count* est un menu permettant de découvrir des gestes basés sur le nombre de points de contact [16, 21]. D'autres exemples de menus gestuels incluent les *Wave* et *Wavelet Menus* [17, 75], les *Leaf Menus* [22, 181], les *ArchMenus* et les *ThumbMenus* [108].

2.3.5 Transition de novice à expert

Un geste peut être extrêmement rapide à exécuter. Par exemple, sélectionner un élément à l'aide d'un *Marking Menu* à un seul niveau prend une fraction de seconde (∼ 175ms d'après Appert et Zhai [7]) car il suffit de tracer un petit trait rectiligne. Appert et Zhai comparèrent ainsi l'utilisation des gestes et des raccourcis clavier [7] avec un guide équivalent (statique indirect, voir 2.3.4). Ils montrèrent que les raccourcis gestuels permettent des temps d'exécution du même ordre que ceux qu'on obtient avec des raccourcis clavier. Toutefois, ils se mémorisent plus rapidement et le temps d'exécution minimal est donc atteinte avec une durée d'apprentissage plus courte.

Les gestes sont toutefois moins faciles à apprendre et à exécuter que les interactions reposant uniquement sur du pointage où un utilisateur a juste à repérer une cible. Cependant, grâce aux guides ils sont accessibles aux utilisateurs novices.

Le défi de ces guides est toutefois de se rendre petit à petit inutiles lorsque l'utilisateur se familiarise avec le système. L'objectif est d'accompagner la transition

entre le mode où l'exécution du geste est exécutée en suivant un guide, appelé mode novice et dans lequel les performances sont lentes, au mode où l'exécution du geste est en mémoire, appelé mode expert et où les performances sont beaucoup plus rapides [242]. Contrairement à la combinaison traditionnelle des menus linéaire et des raccourcis clavier qui proposent un gain de performances extra-modale, les guides permettent un gain de performance intra-modale [59]. En effet, pour compenser les performances limitées des menus linéaires et gagner en productivité, un utilisateur doit de lui-même chercher à apprendre une nouvelle modalité : les raccourcis clavier. Ce changement de modalité provoque paradoxalement un déclin temporaire des performances [196] et, en pratique, est rarement et tardivement effectué [52, 135, 143]. Les guides pour gestes, au contraire, peuvent être vus comme un tutoriel pour le mode expert. Ils répondent en général au principe de conception « Révélation, Guidage et Répétition » introduit par Kurtenbach et al. [132] :

Révélation Le système doit interactivement révéler les commandes disponibles et suggérer la manière de les invoquer,

Guidage Cette révélation doit guider un utilisateur à exécuter une commande dans toute situation,

Répétition Ce guidage doit constituer une répétition de la manière dont un expert exécuterait la commande.

Les *Marking Menus* sont sans doute le premier exemple de technique d'interaction répondant à ces principes. Les éléments graphiques du mode novices (Figure 2.11a) affichent les différentes possibilités (révélation), leur position indique le chemin à tracer (guidage) et ce chemin est exactement celui qui plus tard pourra être suivi en mode expert (Figure 2.11b) sans affichage des éléments graphiques (répétition). En utilisant le mode novice, les utilisateurs apprennent donc petit à petit le mode d'emploi du menu en mode expert (répétition). Ces principes, mis en œuvre avec les *Marking Menus*, ont permit d'atteindre un taux de transition de novice à expert élevé [128].

Certains utilisateurs ont tendance à ne jamais cesser d'utiliser guides, n'atteignant donc jamais les performances optimales [3, 59, 60]. Rendre l'utilisation du mode novice plus difficile peut alors permettre d'améliorer la transition du mode novice au mode expert. Par exemple, Anderson et Bischof proposèrent un guide gestuel, appelé « guide adaptatif », qui disparaît progressivement au fur et à mesure de ses invocations [3]. Il force donc l'utilisateur à mémoriser petit à petit le geste.

2.3.6 Freins à l'adoption des gestes

Aujourd'hui, bien que les interfaces tactiles soient très répandues, les gestes restent relativement peu utilisés pour interagir et les interfaces tactiles reposent principalement sur du pointage. Ceci peut s'expliquer par plusieurs facteurs.

Qwertynomie

La qwertynomie (l'azertynomie dans le monde francophone) illustre de manière particulièrement frappante le fait qu'une solution sous-optimale, une fois mise en place, peut faire obstacle à l'adoption d'une solution plus efficace [65]. Le nom « qwertynomie » vient de la disposition des touches du clavier QWERTY, qui est très largement majoritaires dans le monde anglophone. L'arrangement QWERTY fut originellement conçu pour s'adapter aux contraintes mécaniques des machines à écrire. Il ne fait aucun sens à l'ère des claviers numérique. En fait, il existe des dispositions de touches spécialement optimisées pour la vitesse d'écriture ayant des performances largement supérieures. L'arrangement Dvorak, par exemple, permet de frapper de 20 à 40% plus rapidement [65]. Toutefois, la grande familiarité des utilisateurs avec l'arrangement QWERTY forme une barrière au Dvorak qui, malgré ses avantages, n'a jamais percé.

Le même type de résistance au changement est susceptible de s'appliquer aux interfaces gestuelles et aux gestes [242]. Toutefois, avec l'arrivée de l'informatique mobile, l'industrie informatique montre petit à petit un désir de bousculer les paradigmes d'interaction WIMP dans laquelle elle s'est ancrée depuis les années 80. Ceci peut être l'occasion pour les interfaces gestuelles de percer.

Apprentissage

Contrairement au pointage, les gestes nécessitent un apprentissage et sont donc moins accessibles, ce qui retarde leur adoption [242].

Compatibilité

Enfin, pour éviter un rejet massif de la part des utilisateurs, l'industrie informatique doit s'efforcer d'effectuer un changement de modalité en douceur [242]. Ainsi, il est difficile de supprimer brusquement l'ensemble des éléments interactifs auquel les utilisateurs sont habitués. Une manière de faire est de promouvoir les gestes comme un raccourci accompagnant un menu traditionnel [7, 185].

2.4 Conclusion du chapitre

Pour récapituler ce chapitre, les tablettes tactiles ont le potentiel d'une interaction plus directe, au sens de Shneiderman [201]. Elles peuvent donc permettre une interaction bien adaptées aux utilisateurs novices. Cette propriété ainsi que leur format mobile sont sans doute les principales raisons de leur succès.

Toutefois, les capacités en entrée d'une tablette sont relativement limitées, notamment en ce qui concerne l'ensemble des tâches de pointage sur lesquelles repose la majorité des interactions que l'on trouve aujourd'hui dans les applications commerciales. En effet, leur format réduit entraîne une mauvaise précision sur les cibles de

petite taille (le « problème du gros doigt »), donc la nécessité d'utiliser des objets graphiques plus imposants et donc de réduire leur nombre. De plus, alors que la souris dispose de deux états de pointage actifs (voire davantage, selon le nombre de boutons), une surface tactile n'en possède qu'un seul. Ceci rend notamment les changements de modes plus fastidieux car, hormis les coordonnées sur la surface, assez peu d'information peut être utilisée pour déclencher le basculement de l'un à l'autre.

Les capacités réduites du pointage pourraient être compensées par l'utilisation des gestes. Les surfaces à écran tactile, périphériques absolus et directs, y sont en effet particulièrement propices. Les gestes ont un pouvoir expressif beaucoup plus important que les opérations de pointage. En fait, un pointage peut être vu comme le geste le plus minimal pouvant être capturé par une tablette. De plus, si le geste ne dépend pas d'une position en particulier sur la tablette, il est alors possible de le réaliser sans regarder ce qui prend tout son sens sur des dispositifs mobiles (on parle d'interaction *eyes-free*). Toutefois, les gestes peuvent êtres complexes à exécuter et nécessitent un apprentissage, contrairement, par exemple, à un menu linéaire traditionnel, ce qui nécessite d'utiliser des guides pour aider les utilisateurs novices à les adopter. Enfin, les gestes constituent un changement important face aux interfaces WIMP traditionnelles qui sont maintenant établies depuis plus de 30 ans. Un phénomène de résistance au changement comme celui qui se produit avec la qwertynomie est donc susceptible de compliquer leur adoption. Les limitations inhérentes au pointage en font toutefois une solution alternative particulièrement prometteuse pour l'interaction avec les dispositifs mobiles. De fait, certains gestes simples ne requérant aucun guidage, comme le pinch de la Figure 2.8, ont été proposés dès la popularisation des interfaces *multitouch*, et leur usage a pu se généraliser sans difficulté.

Deux types d'approches peuvent être envisagées pour améliorer l'expressivité de l'interaction avec une tablette et permettre l'utilisation d'applications riches :

1. Mieux exploiter les capacités en entrée existantes en concevant des techniques d'interactions plus expressives. La sélection rapide d'un plus grand nombre de commandes et la transition d'un mode à un autre constituent le principal défi. Dans ce but, les gestes semblent particulièrement prometteurs à condition d'être associés à un guide à la fois performant et qui accompage les utilisateurs dans l'amélioration de leur performance. Cette approche est celle que nous avons suivie pendant la première partie de cette thèse (voir les chapitres 5 et 3).
2. Étendre les capacités en entrée de la tablette en proposant de nouveaux canaux pouvant être exploités. Nous avons suivi cette approche pendant la seconde partie de cette thèse (voir le Chapitre 6).

3 Modes et délimiteurs

Le changement de mode est un cas particulier de la sélection de commandes qui a pour effet de modifier la manière dont les actions utilisateurs se répercutent sur le système, par exemple passer du mode pinceau au mode gomme dans une application de dessin.

Les modes ne sont pas propre à l'informatique. Beaucoup d'objets, parfois bien antérieurs à l'apparition du premier microprocesseur, exposent un comportement modal [116]. Par exemple, les visseuses sont généralement équipées d'un bouton à deux positions dont chacune est associée à un sens de rotation de la mèche, soit deux modes différents. Il est possible de mésinterpréter la position du bouton et, en conséquence, de visser en voulant dévisser et vice-versa (il s'agit donc d'une erreur de mode).

L'utilisation de modes multiples est pratiquement inévitable dans une interface informatique. La quasi-totalité de ce que nous faisons avec un ordinateur implique une transition de mode d'une manière ou d'une autre [116, 160, 198]. Même l'ouverture d'un menu gestuel comme un *Marking Menu* [126] nécessite de basculer dans un mode spécifique où les gestes de l'utilisateur sur la surface seront interprétés dans le contexte de ce menu.

Dans ce chapitre, nous commencerons par revoir les différentes visions que l'on peut trouver dans la littérature, puis proposerons notre modélisation de la notion de mode (Partie 3.1). Par la suite, nous caractériserons plus précisément les différents types de délimiteurs — c'est-à-dire les éléments utilisés pour basculer d'un mode à un autre — et reverrons les différentes approches qui ont été utilisées pour cela en nous concentrant sur le contexte des tablettes tactiles (Partie 3.2).

3.1 Modes

Dans cette partie, nous commencerons par revoir les différentes interprétations qui ont été données aux modes dans la littérature puis nous nous appuierons sur cette littérature pour présenter notre modèle. Pour finir, nous parlerons de quelques cas remarquables et aborderons les problèmes suscités par les interfaces modales ainsi que les solutions qui ont pu être mises en œuvre pour y répondre.

3.1.1 Les modes dans la littérature

Le concept de mode a été beaucoup discuté dans les années 1980-90, à l'époque où la grande majorité de l'interaction humain-machine reposait sur le clavier [116]. Il ne nous a pas été possible d'extraire de notre revue de la littérature une définition précise des modes : leur interprétation est souvent laissée libre.

Pour illustrer l'absence de consensus sur la définition d'un mode, en 1988, Johnson et Engelbeck distribuèrent lors de la conférence CHI un questionnaire portant sur 32 interfaces de logiciels [117]. Ils demandèrent aux participants de juger si ces interfaces étaient ou non modales, l'interprétation des mots « interface modale » étant laissée ouverte. Les participants, des concepteurs d'IHM et des chercheurs de la communauté CHI, classèrent les différentes interfaces qui leurs étaient présentées de manière hautement variable, ce qui souligne bien le caractère flottant de la notion de mode.

Mode comme contexte d'interprétation

Pour Poller et Garter, il y a un changement de mode chaque fois que l'interprétation de la même entrée change [168].

Notre modèle s'inspire principalement de cette définition.

Modes perçus

Raskin donne une définition qui tente de capturer l'aspect subjectif de la perception des modes [175]. Il précise qu'une interface peut paraître modale pour un utilisateur et non modale pour un autre (une interface étant modale si elle contient des modes). Il déduit de différentes études qu'il n'y a pas de mode (perçu) si le changement d'état fait partie du locus d'attention de l'utilisateur.

Ainsi, un mode délimité par sa représentation graphique, comme un bouton, passe souvent inaperçu car le curseur de la souris fait en général partie du locus d'attention lors de l'interaction avec un ordinateur.

Toutefois, ce n'est pas toujours le cas. Pour l'illustrer, une anecdote observée lors de la présentation d'ouverture à Interact 2015 peut être utile. Le présentateur contrôlait le défilement de son diaporama à l'aide du bouton gauche d'une souris tenue à la main. Cliquer n'importe où sur l'écran permettait de passer à la diapositive suivante. Toutefois, lors du déplacement du curseur, apparaissait également deux boutons : l'un avait exactement le même mode que partout ailleurs sur l'écran (passer à la diapositive suivante) et l'autre permettant de revenir à la diapositive précédente. Le présentateur avait tendance à faire bouger la souris dans sa main pendant qu'il parlait ce qui provoquait le déplacement du curseur qui s'est retrouvé à plusieurs reprises, au-dessus du bouton (virtuel) « précédent ». Lorsqu'il pressait le bouton (physique) de la souris, le diaporama reculait donc au lieu d'avancer. Or son locus d'attention étant la salle et éventuellement la souris (l'objet), mais pas

la position du curseur à l'écran (habituellement peu utile et masqué durant une présentation). Il ne comprenait donc pas la réaction du système jusqu'à ce qu'un membre du public lui fasse remarquer où était le curseur.

Modes spatiaux et temporels

Beaudouin-Lafon fait la distinction entre modes spatiaux et modes temporels [1].
Les modes spatiaux sont définis comme suit :

> Avec les modes spatiaux, l'interprétation des actions de l'utilisateur diffère en fonction de la position du périphérique de pointage.

Les modes dits « temporels » rassemblent l'ensemble des modes délimités d'une autre manière et dont la séparation n'est donc pas visible.
Ils sont définis comme suit :

> Pendant un intervalle de temps (de l'activation du mode à sa désactivation), les actions de l'utilisateur sont interprétées de manière particulière.

Ainsi, alors qu'un mode spatial crée une correspondance évidente entre le mode et le lieu où il se manifeste à l'écran, les modes temporels paraissent uniquement se succéder dans le temps.
On peut toutefois noter qu'un mode spatial est lui aussi actif uniquement pendant un laps de temps donné : par exemple, le mode d'une fenêtre n'est actif que lorsque le pointeur se trouve sur l'espace qui lui est alloué.

Modes bloquants

Une boîte de dialogue est dite « modale » lorsqu'elle bloque toutes opérations autres que les siennes. Une boîte de dialogue dite « non modale » peut être laissée en arrière-plan pendant que d'autres interactions sont effectuées avec le système.
En réalité, les deux boîtes de dialogues impliquent des modes. Un nom éventuellement plus adapté peut être « boîte de dialogue bloquante ». La différence est que lorsqu'une boîte bloquante est affichée, tout l'écran sauf la boîte de dialogue est dans le même mode spatial. Ce mode spatial n'étant souvent pas représenté, on a l'impression que l'interface ne répond plus. Une solution pour éviter ce problème pourrait être de griser la zone inactive[1].

Interfaces non modales

Une interface est dite non modale ou sans mode (*modeless*) si elle ne recourt à aucun mode [189] à l'exception de modes spatiaux visibles.
En réalité, il s'agit d'un léger abus de langage, il y a toujours au moins une interprétation de l'entrée utilisateur et donc au minimum un mode. Une interface

1. http://www-ihm.lri.fr/~mbl/ENS/IHM/ecole-in2p3/Cours/cours2.html#RTFToC60

non modale est plus exactement une interface ne comportant qu'un seul et unique mode (sans compter les modes spatiaux visibles).

3.1.2 Modélisation des modes

Notre modèle des modes reprend et étend la définition de Poller et Garter [168] en s'appuyant, pour plus de précision, sur les notions d'action utilisateur et d'interprétation système. Avant de pouvoir parler de mode, il est donc nécessaire de définir précisément ces deux concepts, mais aussi ceux auxquels ils sont liés.

Action utilisateur

Nous définissons une *action utilisateur* A_u par ce que fait physiquement un utilisateur, notamment avec son corps : par exemple déplacer la souris, presser un bouton, lire un paragraphe, lever le bras, bailler ou même juste penser.

Entrée utilisateur

Nous définissons une *entrée utilisateur* E_u comme l'ensemble de l'information portée par une *action utilisateur* A_u qui est capturée (et éventuellement interprétée) par le système. Il s'agit donc de la partie de l'*action utilisateur* perçue par le système à travers des périphériques d'entrée et des capteurs.

Ainsi, un périphérique d'entrée est associé à l'application [2] $p : A_u \rightarrow E_u$. On peut prendre pour exemple le déplacement de la souris ($s : A_u \rightarrow dX,dY$), la position d'un point de contact sur une surface tactile ($st : A_u \rightarrow X,Y$), la direction du regard (détectée par un *eye tracker*), etc.

On peut noter que l'application $p : A_u \rightarrow E_u$ est surjective [3], mais pas injective [4]. En effet la même entrée utilisateur sera capturée suite à un grand nombre d'actions utilisateur différentes car seulement une petite fraction de l'information portée par l'action utilisateur est détectée.

2. En mathématiques, une application est une relation entre deux ensembles pour laquelle chaque élément du premier (appelé ensemble de départ ou source) est relié à un unique élément du second (l'ensemble d'arrivée ou but). https://fr.wikipedia.org/wiki/Application_(mathématiques)

3. En mathématiques, une surjection ou application surjective est une application pour laquelle tout élément de l'ensemble d'arrivée a au moins un antécédent. Voir la figure 3.1. https://fr.wikipedia.org/wiki/Surjection

4. En mathématiques, une application $f : X \rightarrow Y$ est dite injective ou est une injection si tout élément y de Y admet au plus un antécédent x (par f). Voir la figure 3.1. https://fr.wikipedia.org/wiki/Injection_(mathématiques)

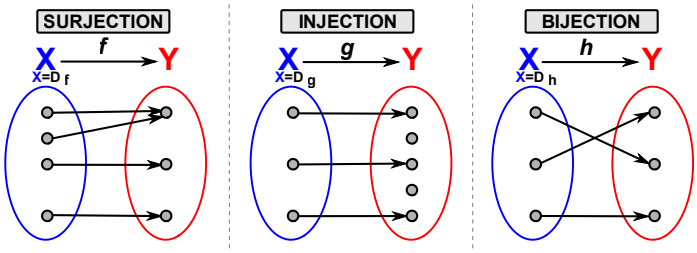


Figure 3.1 – Surjection, injection et bijection (image en provenance de wikipedia).

Réaction système et interprétation système

Nous définissons par *réaction système* R_s la manière dont le système réagit suite à une *entrée utilisateur* E_u. Nous appelons *interprétation système* la relation $I_s : E_u \to R_s$. Par exemple, suite au déplacement de la souris (action utilisateur), et à la détection de ce mouvement par le capteur optique de la souris (entrée utilisateur), le système réagit en général en déplaçant un curseur à l'écran (réaction système).

Il est important de noter que la position du curseur n'est pas une information directement capturée par un dispositif d'entrée. Bien qu'elle soit (en général) modifiée en réaction à une entrée utilisateur, nous ne considérons donc pas ici le curseur lui-même comme une entrée utilisateur. On peut d'ailleurs remarquer que la position de ce curseur est parfois modifiée automatiquement par le système sans aucun rapport avec une action utilisateur. Par exemple, certaines anciennes versions de *Microsoft Windows* pouvaient déplacer automatiquement le curseur sur le bouton par défaut d'une boîte de dialogue lorsqu'elle était affichée.

L'interprétation système I_s est une relation qui n'est ni surjective ni même une application, car la même entrée utilisateur peut provoquer des réactions systèmes différentes selon le *mode* actif. Par exemple lorsque le bouton gauche de la souris est enfoncé, le déplacement de la souris peut être interprété pour sélectionner du texte. Dans un jeu vidéo de tir à la première personne (*FPS*), il sera souvent interprété comme une rotation de la caméra.

Modes

Nous retenons la définition suivante, dont la première ébauche a été proposée par Poller et Garter en 1983 [168] :

> Un *mode* correspond à une *réaction système* donnée suite à une *entrée utilisateur*. Pour chaque réaction système différente suite à la même *entrée utilisateur*, il y a donc un *mode* différent [116, 168, 210, 211].

La touche « verrouillage majuscule », par exemple, permet de basculer entre deux modes. Dans le premier cas, frapper une touche alphabétique du clavier saisit

la lettre associée en minuscule. Dans le second cas, la même action utilisateur saisit cette lettre en majuscule. Les boutons des interfaces graphiques manipulés à travers un périphérique de pointage relatif sont un autre exemple de modes. En fonction de la position du curseur (et plus précisément en fonction du bouton virtuel sur lequel il se trouve), un clic de la souris n'a pas le même résultat. En effet, si le curseur de la souris se trouve sur le bouton correspondant à la commande « enregistrer », un clic de la souris (l'action utilisateur) provoquera alors l'enregistrement d'un document (réaction système). Si au contraire il se trouve sur le bouton correspondant à la commande « quitter », cette fois, un clic de la souris (la même action utilisateur) provoquera la fermeture de l'application ouverte (une réaction système différente).

L'*interprétation système* pour un *mode* M donné $I_s(M) : E_u \rightarrow R_s$ est une surjection : pour une *entrée utilisateur* donnée et un *mode* donné, la *réaction système* est toujours la même

Quasimodes. Nous parlerons de « *quasimodes* » lorsque l'interprétation de l'entrée utilisateur est liée à une action physique continue de l'utilisateur [5] [175]. Une partie de l'entrée utilisateur sert donc de modifieur. Par exemple, maintenir ou non la touche « majuscule » (partie modifieur) modifie l'interprétation de la frappe sur une autre des touches du clavier. En considérant l'action utilisateur dans son ensemble, il ne s'agit pas tout à fait de la même action (dans un cas l'utilisateur maintient la touche « majuscule », dans l'autre non) et donc pas tout à fait un mode : un quasimode. Les interactions de type « *drag and drop* » sont un autre exemple de quasimode : maintenir ou non le bouton de la souris sert à modifier l'interprétation du déplacement de ce périphérique.

En ce qui concerne le système, rien ne différencie vraiment un mode d'un quasimode. Toutefois, d'un point de vue utilisateur, les quasimodes ont des propriétés intéressantes vis-à-vis des modes : en forçant l'utilisateur à maintenir une tension musculaire pour rester actif, ils améliorent grandement la conscience du mode courant et limitent d'autant les erreurs de modes [198]. Nous reviendrons sur ce point dans la Partie 3.1.5.

Les quasimodes sont également parfois appelés *Spring loaded mode* [101, 117], *user maintained mode*[198] ou micro-modes[5].

Délimiteurs

Les délimiteurs sont utilisés pour marquer les séparations entre modes, comme la transition entre la sélection d'un outil et sa manipulation [99, 141]. Par exemple, Hinckley et al. utilisèrent dans Scriboli une *pigtail* (ou « queue de cochon », voir la Figure 3.2) pour séparer, au sein d'un seul et même mouvement, la sélection d'un objet et la sélection d'une commande affectant cet objet [99].

5. http://www-ihm.lri.fr/~mbl/ENS/IHM/ecole-in2p3/Cours/cours2.html#RTFToC60

Figure 3.2 – *Pigtail* dans Scriboli [99]. La détection d'une boucle est utilisée pour ouvrir un *Marking Menu*.

Si l'on représente les différents modes lors de l'interaction avec un système sous la forme d'une machine à état dont chaque état représente un mode, les délimiteurs sont les évènements provoquant les différentes transitions entre modes.

Ils sont donc indispensables à l'utilisation de modes. Il en existe une large variété : il suffit de « quelque chose de différent » dans le flux des entrées utilisateur [99].

On peut toutefois distinguer les types de délimiteurs suivants :

Les délimiteurs spatiaux qui interprètent différemment les actions utilisateurs en fonction de leurs coordonnées dans un espace (généralement l'espace d'affichage). Comme nous l'avons vu, les modes dont la séparation est marquée par un délimiteur spatial sont appelés « modes spatiaux ». Il s'agit typiquement des boutons ou des fenêtres.

Les délimiteurs temporels qui reposent sur le temps. Il s'agit typiquement des délais. Les délimiteurs temporels n'ont que peu à voir avec la définition de mode temporel. Un « mode temporel » au sens de Beaudouin-Lafon n'est pas nécessairement délimité par un délimiteur temporel.

Les délimiteurs gestuels qui reposent sur la reconnaissance d'un ou plusieurs gestes. Les *pigtails* de Scriboli en sont un exemple connu [99].

Les délimiteurs ont une importance capitale puisqu'ils permettent de basculer entre les différents modes. Ils sont de plus parfois associés à un retour utilisateur sur la manière dont s'active un mode : par exemple, la position et la taille d'un bouton (délimiteur spatial) permettent d'indiquer les frontières du mode associé.

Le cas des tablettes tactiles

Sur un ordinateur traditionnel, une souris propose au moins deux états actifs (bouton enfoncé bouton relevé) et un état passif (lorsque la souris est en l'air pendant un geste de *clutching*) [44]. Une manière évidente de le mettre à profit est d'associer à chaque état actif une interprétation différente. Toutefois ce n'est pas systématique, et il ne faut donc pas confondre état du périphérique de pointage et quasimode. Par exemple, le bouton droit de la souris n'est pas toujours pris en

compte, ou pourrait être interprété de la même manière que le bouton gauche. À la souris s'ajoute également l'utilisation du clavier dont certaines touches servent souvent de modifieurs (« ctrl », « alt », « shift » ou « cmd ») pouvant aussi être exploités pour délimiter des quasimodes.

Sur tablette tactile, le dispositif d'entrée principal, la surface tactile, dispose également de deux états : lorsque l'utilisateur touche l'écran ou non [44]. Toutefois, le système reçoit de l'information seulement dans un seul de ces états : lorsque l'utilisateur touche l'écran. Dans le second, lorsque l'utilisateur ne touche pas l'écran, rien n'est capturé. De plus, l'absence de clavier physique (dans la plupart des cas) ne permet pas d'utiliser des modifieurs au clavier. L'exploitation du *multitouch* est une solution [16, 21] pour augmenter le nombre d'états, mais davantage de surface d'affichage est alors occultée et la position de référence est ambigüe (puisqu'il y a plusieurs contacts). Ainsi, il est plus difficile de mettre en place des quasimodes [101]. Un concepteur doit donc en général recourir à davantage de modes (à part entière) et bien souvent à des techniques de sélection pour passer de l'un à l'autre. Aujourd'hui sur tablette tactile, un utilisateur change en général de mode à l'aide de boutons sur une barre d'outils ou à travers des menus linéaires si le nombre de boutons nécessaires est trop élevé pour en permettre l'affichage permanent.

D'après Thimbleby, l'utilisation de modes est une méthode pour accroître la bande passante interactionnelle limitée des dispositifs d'entrée [210]. Le nombre d'actions systèmes possibles (par exemple déplacer une image, dessiner, sélectionner du texte) excédant largement le nombre d'actions utilisateurs pouvant être détectées par le système (par exemple déplacer la souris), il est nécessaire d'associer à ces dernières différentes interprétations. Les capacités interactives des tablettes tactiles étant réduites comparativement à un ordinateur traditionnel (petite surface d'affichage, précision moindre, moins d'états de pointage, etc., voir le Chapitre 2), elles sont particulièrement exposées à cette difficulté [101]. Notre hypothèse, est que l'absence de méthode rapide pour passer d'un mode à un autre sur tablette force les concepteurs à restreindre le nombre de fonctionnalités dans leurs applications.

3.1.3 Cas remarquables

Pour rappel, nous avons défini un mode comme suit :

> Un *mode* correspond à une *réaction système* donnée suite à une *entrée utilisateur*. Pour chaque réaction système différente suite à la même *entrée utilisateur*, il y a un *mode* différent [116, 168, 210, 211].

Dans cette partie, nous revenons sur quelques cas intéressants, conséquences de cette définition.

Segmentation des gestes

L'interprétation du mouvement est parfois amenée à changer au sein d'un même geste. Par exemple, les *Control Menus* [169] sont des *Marking Menus* [126] qui fusionnent la sélection de commandes et le contrôle continu (voir la Partie 3.2.1 page 56). Le geste est donc décomposé en deux parties :

1. La direction initialement empruntée par le geste détermine le choix du paramètre à modifier ;
2. après une certaine distance, le réglage du paramètre en question peut commencer.

Ces deux parties correspondent clairement à deux états systèmes différents et deux contextes d'interprétation de l'information reçue par la surface tactile différents.

Deux approches sont possibles pour interpréter l'action utilisateur :

1. considérer l'action utilisateur à un temps T quel que soit les événements précédents,
2. considérer l'action utilisateur dans son ensemble et en particulier, en tenant compte du temps et des évènements passés.

Par commodité, nous tranchons pour la première option. La seconde introduit en effet une ambiguïté : comment segmenter un geste ? Où commence et où s'arrête une action donnée ?

Modes versus paramètres

Certains paramètres entraînent un changement de mode.

Prenons l'exemple d'une application de dessin où la couleur du pinceau peut être réglée parmi les trois couleurs primaires à l'aide de boutons dans une barre d'outils. Dessiner avec chacune de ces couleurs associe à la même action utilisateur un comportement différent (dans un cas un trait bleu, dans l'autre un trait rouge, etc.). Cela peut donc être vue comme trois modes distincts selon notre définition. Au lieu de couleur, il aurait pu s'agir d'outils, par exemple pinceau, pot de peinture ou création de cercles.

Un corollaire de cette propriété est que certains réglages sont associés à une quasi-infinité de modes. Imaginons qu'au lieu de sélectionner la couleur en utilisant les trois boutons de la barre d'outils, un utilisateur se serve d'une pipette à couleur. Cette dernière permet, si on ignore les limitations intrinsèques à la mémoire informatique et à la précision du dispositif d'entrée, de choisir parmi une infinité de teintes différentes. À chacune de ces teintes correspond alors un mode différent.

3.1.4 Préemption

Les modes, à l'exception des modes spatiaux visibles, furent longtemps fortement découragés [8, 101, 151, 175, 209]. Par exemple, les directives de conception publiées

par Apple en 1992 conseillaient de « recourir essentiellement à des fonctionnalités *modeless*, qui autorisent les gens à faire tout ce qu'ils veulent quand ils le veulent avec l'application » [8]. Aucune définition précise d'un mode n'est toutefois donnée dans le document sinon qu'un mode « restreint généralement les opérations qu'un utilisateur peut effectuer tant qu'il est actif ». Cela laisse donc penser qu'ils utilisent la définition des modes bloquants présentée en 3.1.1. Toutefois, également, Tesler est connu pour sa célèbre croisade : « *Don't mode me in!* » et décrit un mode comme « un état de l'interface utilisateur qui dure une période de temps donnée, qui n'est associé à aucun but particulier et n'a aucun rôle sinon celui de démarquer une interprétation des dispositifs d'entrée » [209], ce qui s'approche de notre définition.

Le reproche majeur qui est fait est de bloquer les utilisateurs qui peuvent s'en sentir prisonniers. Il s'agit du problème de préemption. Par exemple, le logiciel de traitement de texte en ligne de commande Vi (et son successeur Vim) est connu pour être difficile à maîtriser notamment en raison de son usage de modes. Quitter le programme nécessite en effet de passer en mode commande, de taper la lettre « q » puis de valider. Passer en mode commande nécessite d'être dans le mode visualisation et de presser la touche « : ». Retourner dans le mode visualisation depuis le mode insertion (utilisé pour écrire) nécessite de presser la touche « Esc ». (la Figure 3.3 traite ce problème avec humour).

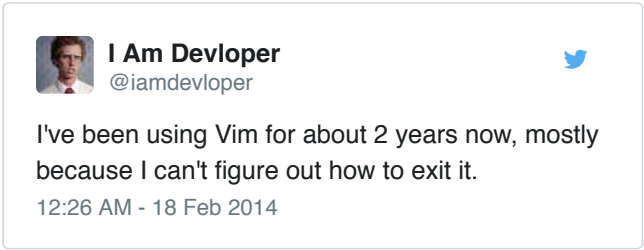


Figure 3.3 – « J'utilise Vim depuis environ deux ans maintenant, essentiellement parce que je ne trouve pas le moyen d'en sortir. » Citation de I Am Devloper sur Twitter.

En réalité, Vi expose peu de modes : commandes, visualisation et édition. Sur une interface WIMP, rien qu'une barre d'outils peut être composée de plus d'une vingtaine de boutons et donc une vingtaine de modes délimités par la représentation graphique de ces boutons. Ainsi, plusieurs auteurs notent que les modes sont souvent moins la source du problème que le modèle interactif lui-même [116, 190]. Notre hypothèse est que la difficulté avec Vi n'est pas l'utilisation de modes, mais bien plutôt les délimiteurs qui sont utilisés pour passer de l'un à l'autre : ces délimiteurs sont tous différents, propres à Vi (peu d'autres logiciels utilisent ces conventions) et doivent donc être appris et mémorisés.

Selon Tesler, les fenêtres, introduites avec les interfaces WIMP, ne posent

pas de problème de préemption [209]. Pourtant, selon les termes de Tesler lui même, elles ne sont que « des modes déguisées en agneaux ». Le principal atout des fenêtres est qu'elles donnent un délimiteur standard reposant sur une métaphore graphique claire pour passer d'un mode à un autre. L'importance d'un délimiteur standard a été remarquée par Thimbleby [210]. Thimbleby maintient qu'une fonction *reset* permettant de restaurer un mode de fonctionnement familier d'où il est possible, le cas échéant, de reprendre ses marques est souvent essentiel [210]. On peut d'ailleurs remarquer que sur les tablettes et téléphones Apple, le bouton *home* permet toujours de retourner immédiatement sur l'écran d'accueil (sauf lorsque la tablette est allumée). Toutefois les fenêtres ne peuvent pas s'appliquer partout : par exemple, il semblerait peu judicieux de répartir les différents modes de Vi sur différentes fenêtres.

3.1.5 Erreurs de mode

La préemption n'est pas le seul inconvénient des modes. Une autre difficulté avec Vi est qu'il est malaisé d'identifier, à un instant donné, quel est le mode actif. Ceci est donc source d'erreurs appelées « erreurs de mode » [159, 168, 211]. Norman en donne la définition suivante :

> « Les erreurs de modes se produisent quand un utilisateur exécute une action qui est appropriée dans certaines situations, mais qui est inappropriée dans la situation présente » [159].

Le système réagit alors de manière inattendue pour l'utilisateur. Par exemple, il est facile de chercher à insérer un paragraphe sans s'apercevoir que l'éditeur est en mode « remplacer » (et non pas en mode « insérer ») ce qui va se traduire par une portion de texte effacée par inadvertance. Malgré le voyant lumineux associé au verrouillage majuscule, il est également courant de vouloir écrire en minuscules alors que le mode majuscule est actif. Ceci pose particulièrement problème pour les champs de mots de passe où les caractères sont cachés pour des raisons de sécurité. Afin d'éviter la frustration d'une saisie systématiquement erronée, un message d'avertissement est dans ce cas parfois affiché à côté des champs de mots de passe (voir la Figure 3.4).

Limiter les erreurs de modes

Une méthode pour se prémunir des erreurs de modes consiste à équiper l'interface avec des indicateurs clairs du mode courant [154, 160, 198, 211, 237]. La mise en place d'un retour utilisateur efficace n'est toutefois pas aisée. Le retour doit à la fois être difficile à ignorer [116] sans toutefois nuire à l'interaction et détourner l'utilisateur de sa tâche.

Les modes spatialement délimités proposent en général un retour visuel efficace sur la présence d'un mode [105, 154, 209], et sur son état (le mode est actif si

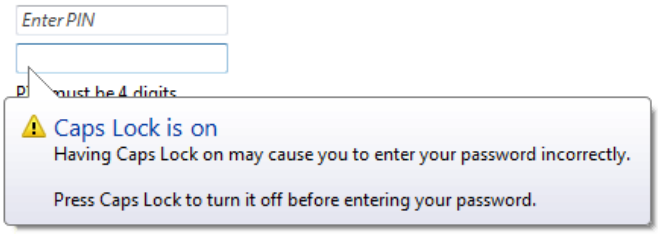


Figure 3.4 – Message d'avertissement lors de l'entrée d'un mot de passe

le curseur est au-dessus de la zone d'affichage réservée au mode, par exemple un bouton). Ils consomment toutefois plus ou moins d'espace d'affichage, ce qui en complique l'usage sur un dispositif mobile. De plus, l'écran est souvent encombré, de sorte que le retour utilisateur devient plus difficile à remarquer, notamment lorsque l'utilisateur est concentré sur son objet d'intérêt [198].

L'utilisation de curseurs spécifiques [116, 189, 211] dans l'entrée de texte ou le déplacement de la souris [154] est un autre type de solution. Par exemple, dans un navigateur internet, le curseur prend en général la forme d'une main pour symboliser que l'action de cliquer va ouvrir un lien. Toutefois, le curseur n'est pas affiché sur les tablettes tactiles qui se reposent sur un pointage en contrôle absolu direct (voir la Partie 2.2.1).

Le retour audio a été également utilisé, notamment lors de la frappe au clavier [154] ou avec des icônes sonores (la commande associée à un mode, délimité par exemple avec un bouton, est symbolisée par un son) [43].

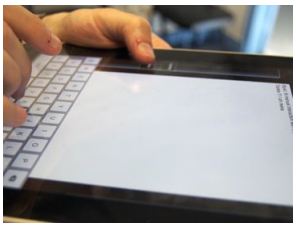
Enfin, une autre méthode est de demander à l'utilisateur de maintenir une tension musculaire pour garder un mode actif [101, 198], par exemple, en lui demandant de maintenir une touche pressée (comme la touche « majuscule » pour passer en mode majuscule). On obtient donc un retour proprioceptif. Il s'agit en fait de mettre en place un quasimode. Wagner et al. ont étudié la manière dont est tenue une tablette et proposèrent des boutons virtuels manipulés avec la main qui tient l'appareil [222], explorant entre autres comment créer des quasimodes avec ces boutons (voir la Figure 3.5). Leurs travaux ont été ensuite repris par Foucault et al. [74].

Contrôler les conséquences

Les erreurs de modes n'ont pas toujours les mêmes conséquences. Alors que dans le cas d'une arme à feu, interpréter à tort que la sécurité est active peut s'avérer dramatique, avec une page Web, cliquer alors que le curseur ne se trouve pas sur le lien cible n'a bien souvent aucun effet.

Pour généraliser, Thimbleby affirme qu'une erreur de mode a peu d'importance si les modes ne se chevauchent pas. Deux modes ne se chevauchent pas si l'inter-

(a) Navigation dans un PDF



(b) Passer en majuscule

(c) Zoomer dans une carte

Figure 3.5 – Bipad : technique de quasimode bi-manuelle pour tablette [222]

prétation de l'entrée utilisateur valide dans un mode est toujours invalide dans un autre. On perd toutefois une grande partie de l'intérêt des modes, c'est-à-dire le pouvoir de faire jouer plusieurs rôles à un seul et même dispositif d'entrée [210].

La possibilité de revenir en arrière permet à un utilisateur d'annuler les conséquences d'une action erronée. Cette fonctionnalité est toujours recommandée, car elle évite beaucoup de frustration [160].

Charge cognitive

Certains travaux indiquent que même si un utilisateur commet peu d'erreurs, les modes impliquent une charge cognitive supplémentaire, en particulier quand les transitions entre modes peuvent être déclenchées automatiquement par le système ou par d'autres personnes [154, 190, 237]. En effet, lorsqu'un mode peut être modifié sans intervention de l'utilisateur, ce dernier doit apprendre à se servir de tous les modes et à identifier, avant toute action, le mode actif [190, 237].

Il s'agit toutefois d'un cas particulier. Les modes peuvent également permettre de mieux séparer les différents contextes d'interaction. Par exemple, comme noté par Johnson, certains modes peuvent n'être activés qu'occasionnellement, ce qui permet de pousser hors de l'attention de l'utilisateur tout un pan des fonctionnalités du système qui ne sont utiles que dans quelques cas bien précis [78, 116]. On peut donc considérer qu'à un instant donné il y a moins d'options possibles, ce qui devrait se traduire, selon loi de Hick [98], par un gain de productivité [211].

Modes comme limitateurs d'erreur

Bloquer les fonctionnalités risquées ou impossibles en l'état est aussi une manière d'éviter certaines erreurs [116, 160]. Johnson [116] donne ainsi l'exemple de la sécurité des armes à feu. Il s'agit d'un mode dont le but — rendre l'objet purement et simplement inopérant le plus souvent possible — pourrait paraître frustrant si les conséquences d'une mauvaise manipulation n'étaient pas aussi dramatiques potentiellement.

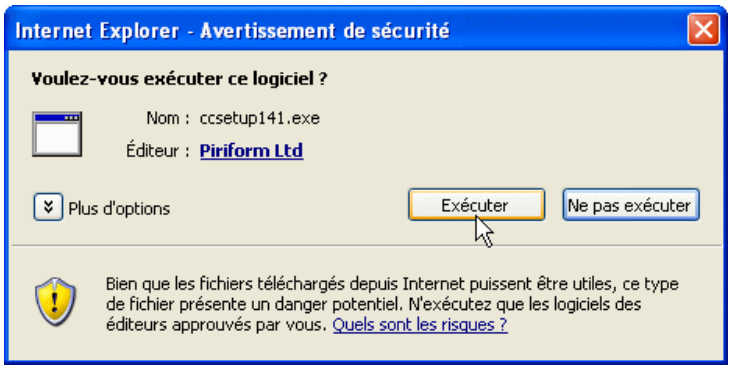


Figure 3.6 – Boutons : un exemple de délimiteur spatial. Selon la position du curseur, un clic de la souris ne provoque pas la même réponse système (ici, exécuter « ccsetup141.exe », ne pas exécuter et fermer la fenêtre, afficher plus d'options, etc.)

3.2 Délimiteurs

Nous appelons *délimiteur* un élément remarquable au sein du flux d'entrée utilisateur propre à un mode. Il s'agit de « quelque chose de différent » [99] qui peut être reconnu et exploité. Nous appelons *modes sources* d'un délimiteur les modes dans lequel ce délimiteur est interprété en tant que tel.

Les délimiteurs sont notamment utilisés pour marquer la séparation entre les modes ou les quasimodes, par exemple pour définir la transition entre un mode de sélection et de manipulation d'outil [99, 141]. Par exemple, Hinckley et al. ont utilisé dans Scriboli une *pigtail* (ou « queue de cochon ») pour séparer la sélection d'objets et la sélection de commandes au sein du même mouvement [99]. Les boutons des interfaces graphiques (voir la Figure 3.6) sont un autre exemple de délimiteur qui repose sur les coordonnées du curseur de la souris.

Les délimiteurs sont aussi exploités pour délimiter l'exécution de commande sans provoquer nécessairement un changement de mode. Par exemple les touches alphanumériques d'un clavier physique sont des délimiteurs qui activent la saisie du caractère associé. Positionner le curseur sur un bouton graphique délimite la transition vers un mode conditionnant l'interprétation du clic de la souris. L'action de cliquer, au contraire, délimite au sein de ce mode l'activation de la commande associée au bouton, sans provoquer de transition de mode en elle-même, sauf cas particuliers.

Dans cette partie, nous commencerons par une revue, dans le cadre de la sélection de commande, des différents contextes dans lesquels les délimiteurs sont mis en œuvre. Nous proposerons ensuite une classification des délimiteurs en nous intéressant plus particulièrement à ceux qui sont exploitables sur les tablettes tactiles.

3.2.1 Contexte de délimitation

Dans le cadre de la sélection de commande, il est utile de distinguer trois contextes de délimitation : (1) L'activation de la technique de sélection, la, (2) segmentation des différents modes mis en jeux au sein même de la technique de sélection, et la, et (3) terminaison. Dans chacun de ces contextes, des caractéristiques différentes vont influencer le choix d'un délimiteur particulier.

Activation

L'activation de la technique est une des délimitations les plus critiques. La particularité du délimiteur d'activation est qu'il a lieu avant que la technique d'interaction ne prenne la main, puisque c'est lui qui permet de l'enclencher. Il doit donc s'accommoder des autres interactions possibles avec le système alors que, une fois entré dans un mode propre à la technique d'interaction, il lui devient possible d'intercepter l'ensemble des actions utilisateurs.

Dans le contexte des gestes, la validation ou l'invalidation de la reconnaissance d'un délimiteur d'activation nécessite en règle générale une étape (associée à un mode) d'enregistrement (*registration*) [76, 224, 229, 239]. Son objectif est de recueillir suffisemment d'actions pour permettre au reconnaisseur de prendre une décision concernant l'activation ou non de la technique.

Beaucoup d'applications permettent des interactions continues comme le défilement et le zoom [141]. Par exemple, avec un logiciel de peinture sur tablette, l'utilisateur s'attend à dessiner dès que le doigt touche la surface. Une action de ce type est difficile à délimiter, car elle a vocation à exploiter l'ensemble du mouvement et doit prendre effet quasi instantanément, ce qui laisse peu de marge pour un délimiteur d'activation.

La littérature sur les techniques de menus gestuels ignore souvent la problématique du délimiteur d'activation. Par exemple, les *Marking Menus* [125, 126] sont activés dès que l'utilisateur touche la surface, ce qui ne laisse place à aucune interaction continue. En conséquence, un concepteur d'application souhaitant les utiliser devra lui même mettre en place son propre délimiteur afin de faire basculer le système dans un mode dédié au menu (à moins que le menu ne soit la seule interaction tactile possible, ce qui est rarement le cas). Laisser à la charge du concepteur d'application le soin de définir un délimiteur peut être justifié par le fait que l'inventeur d'une technique n'a pas connaissance du contexte dans lequel la technique va être mise en œuvre. Toutefois, la prise en compte de ce facteur dès la conception peut permettre une meilleure intégration avec la technique. De plus, certains délimiteurs comportent des degrés de liberté non utilisés qui peuvent être exploités pour améliorer l'expressivité de la technique. Par exemple, nous présenterons en 4.2 les *SigmaMenus*, un nouveau menu gestuel inspiré des *Marking Menus* et qui sont activés par un mouvement circulaire. La sélection prend en compte le sens de

rotation de ce mouvement pour doubler le nombre de commandes accessibles.

Parmi les papiers s'intéressant aux délimiteurs d'activation, de nombreux travaux proposent d'utiliser un délimiteur différent pour chacun des modes disponibles [76, 96, 144, 166, 238]. Il n'y a donc pas de menu. Par exemple les *TouchTools* de Harrison et al. [96] (Figure 3.10) permettent de sélectionner un mode de manipulation en fonction de la posture de la main. Avec cette méthode, l'entrée dans chaque mode doit être mémorisée séparément et l'utilisation d'un guide (voir la Partie 2.3.4) est difficile puisqu'il n'y a pas de mode dédié à la sélection elle-même. De plus, le nombre de délimiteurs compatibles avec le mode par défaut d'une application — et donc le nombre de modes accessibles de cette manière — est souvent limité.

Segmentation

Les délimiteurs de segmentation servent à alterner entre les différents modes internes à la technique de sélection. Par exemple, les *Marking Menu* [125, 126] sont composés de deux modes principaux : le mode novice et le mode expert. Le mode expert est le mode de départ, enclenché après l'activation de la technique. Le mode novice est délimité par une pause statique de quelques centisecondes (le doigt reste sur la surface sans bouger). De plus, en mode novice et si le *Marking Menu* est hiérarchique, les sous-menus s'ouvrent également à la suite d'un léger délai après s'être déplacé dans la direction de l'élément qui le représente. Ce délimiteur pourrait être remplacé par le franchissement d'une distance par exemple. Ceci pourrait permettre d'accélérer l'exécution du mode novice, mais le geste deviendrait sensible à l'échelle et l'insensibilité à l'échelle est une caractéristique chère à leur inventeur [131].

Les *Control Menus* (voir la Figure 3.7) [169] et les *FlowMenus* [90] regroupent au sein d'un seul tracé, sans relever le doigt, la sélection et le contrôle continu. Ces deux étapes se confondent donc en un seul et même mouvement, ce qui est susceptible d'améliorer les performances [88]. À terme, le regroupement peut devenir un automatisme et le geste peut être exécuté comme un tout ce qui présente des avantages cognitifs [45, 88]. Raskin note toutefois que ce type d'automatisme (*chunking* selon les termes de Buxton [45]) peut s'observer mais avec des gestes comportant plusieurs tracés en surface [175]. Il est probable que le critère principal soit plutôt un délimiteur le moins intrusif possible. Ainsi relever et reposer le doigt reste assez peu coûteux cognitivement, peut-être moins qu'une *pigtail* qui ne nécessite pas de relever le doigt.

Terminaison

Le délimiteur de terminaison marque le moment où la technique d'interaction rend la main au système (et donc en général au mode par défaut). Sur tablette tactile, le fait de relever le doigt de la surface marque souvent la fin d'une interaction

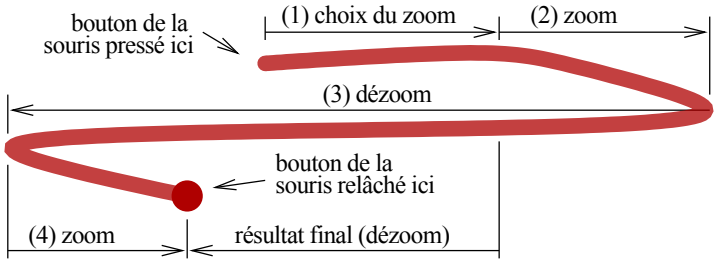


Figure 3.7 – Trace correspondant à la sélection et à la manipulation du zoom à l'aide d'un *Control Menu* [169]. Le délimiteur de segmentation correspond à la distance parcourue en (1).

Toutefois, cela n'est pas toujours adapté [100], comme par exemple dans le cas où la technique d'interaction est composée de plusieurs traits comme les *Multi Stroke Marking Menus* [240, 244]. Sur l'application mobile *Google Maps*, il est possible d'entrer avec un tap dans un mode permettant de zoomer et de dézoomer mais ce mode est immédiatement quitté après un court délai si son activation n'a pas été immédiatement suivie d'un nouveau contact avec la surface.

Délimiteurs a posteriori

Zeleznik et Miller ont proposé un délimiteur modifiant l'interprétation non pas des actions à venir, mais du geste qui l'a précédé [240]. Par défaut les traits sur la surface permettent de dessiner. S'ils sont ponctués d'un tap, ils sont alors interprétés comme une commande gestuelle.

3.2.2 Classification des délimiteurs

Il est difficile de définir complètement l'espace des délimiteurs. Comme on l'a déjà noté, il suffit de « quelque chose de différent » dans le flux des entrées utilisateur [99]. Un grand nombre de dimensions peuvent donc être exploitées. Cet état de l'art ne peut donc prétendre à l'exhaustivité bien que nous ayons tenté de couvrir autant que possible les différents types de délimiteurs. On se concentrera dans cette classification sur les délimiteurs utilisables ou pouvant être adaptés aux tablettes tactiles.

Freeman et al. ont proposé une taxonomie des délimiteurs d'activation dans le contexte des gestes de surface [76]. Cette taxonomie comporte les axes suivants : Délimiteurs à un doigt, délimiteurs *multitouch*, délimiteur à une forme, délimiteur multiforme. On s'intéressera ici à une classification un peu plus large, sur l'ensemble des délimiteurs possible sur tablettes tactiles en s'appuyant sur notre définition d'un mode. On élargira sur les délimiteurs sur d'autres plateformes, par exemple

avec les gestes dans l'air, qui peuvent aussi apporter des solutions intéressantes dans notre contexte.

Il est important de noter que les catégories de délimiteurs présentées ici ne sont pas orthogonales : un délimiteur recouvre souvent plusieurs de ces dimensions. Par exemple, Johnson et al. ont proposé de cliquer sur un objet graphique suffisamment longtemps pour activer un mode d'édition [115]. Il s'agit donc d'un délimiteur à la fois spatial et temporel.

Délimiteurs spatiaux

Il s'agit typiquement de boutons, par exemple, ceux d'une barre d'outils. Le mode est délimité par une zone de l'écran.

Par exemple, *FastTap* est un menu que l'on ouvre avec un bouton situé dans le coin inférieur gauche et qui se déploie sur l'ensemble de la surface d'affichage [91]. Le bouton pour ouvrir le menu est un délimiteur spatial. Une fois ouvert, chacun des boutons à l'intérieur du menu est également un délimiteur spatial. Il en va de même pour n'importe quel menu traditionnel. Hinckley et al., dans le cadre de Scriboli [99], ont étudié l'utilisation d'une poignée pour distinguer la sélection d'objets de l'ouverture d'un *Marking Menu*. La poignée consiste en une zone rectangulaire apparaissant juste après un lasso de sélection d'objets à l'endroit où le stylet a quitté l'écran (Scriboli était prévu pour le stylet). Si l'utilisateur repose le stylet en dehors de cette zone, il peut ajouter de nouveaux objets à sa sélection. S'il le repose dans la poignée, le *Marking Menu* s'active et il peut donc poursuivre en sélectionnant une commande. BiPad [222] (Figure3.5) est une petite barre d'outils affichée sur le côté droit de la tablette et qui s'utilise avec la main qui porte le dispositif. Chacun des boutons de cette barre d'outils est un délimiteur spatial. SPad [74] étend le principe de BiPad en rajoutant une zone, entre la barre d'outils et le cadre de l'écran, où un *Marking Menu* peut être exécuté afin de modifier les commandes disponibles.

Les fenêtres (Figure 3.8) sont aussi délimitées par la position de leur représentation à l'écran [105, 209]. Similairement les *Zone Menus*, les *Polygon Menus* [243] et la *Hotbox* [129] divisent l'espace d'affichage en plusieurs parties, chacune associée à un *Marking Menu* [126] différent.

Souvent, les objets de la scène peuvent eux-même permettre d'activer un menu ou une technique [115, 220]. Par exemple lors de la sélection d'objets avec le doigt, *Shift* [220] détermine si la cible est cachée par la taille du doigt et, dans ce cas, affiche une loupe permettant d'affiner la sélection en contrôle indirecte. Également, Johnson et al. ont permis l'édition de dessins à main levée avec un appui long sur le dessin (délimiteur à la fois spatial et temporel).

Une *Toolglass* est une barre d'outils semi-transparente qui permet de sélectionner une commande depuis une palette sur un objet d'intérêt en cliquant « à travers » la barre d'outil sur l'objet [35, 36, 130]. Les différents modes possibles sont donc

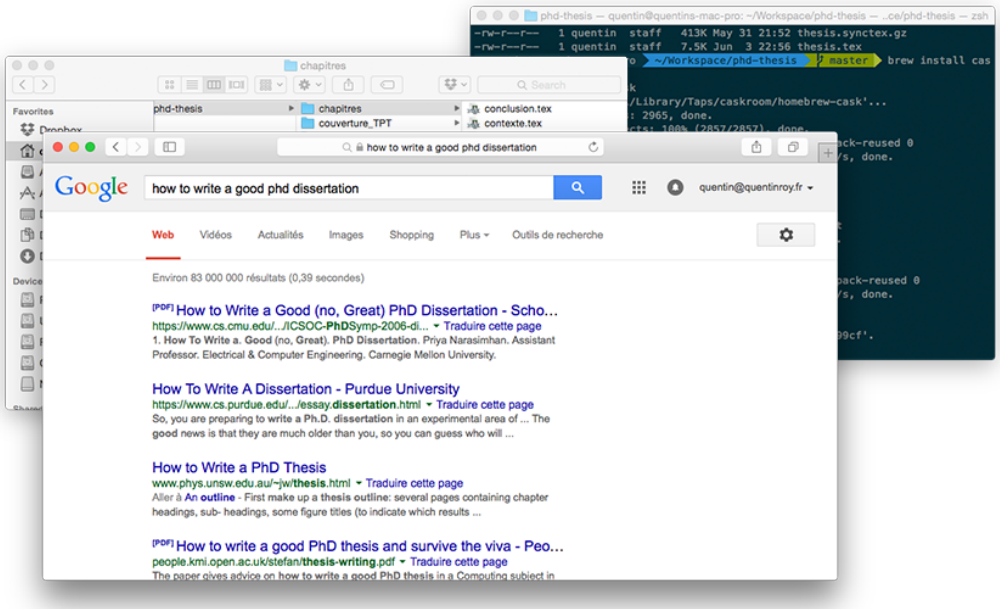


Figure 3.8 – Fenêtres : une forme d'interaction modale [105, 209]

délimités par la position de la barre d'outils et la position des objets graphiques.

Les bords de l'écran sont une position de choix pour un délimiteur [39, 113, 179]. L'idée est de démarrer le geste en dehors de l'écran puis de le faire entrer. On retrouve cette technique pour ouvrir un panneau latéral dans certaines applications commerciales sur mobile. Ceci rejoint le principe du franchissement [1, 4, 5], c'est-à-dire le fait non pas de pointer sur une zone en particulier, mais de la traverser. Par exemple, le passage sur une case à cocher peut changer son état (voir la Figure 3.9). L'idée est de se reposer uniquement sur la position spatiale pour délimiter une action, sans se préoccuper des états du pointage et de leurs transitions (un clic de la souris ou un tap du doigt sur une tablette n'est donc pas pris en compte).

Les *Control Menus* [169] et les *FlowMenus* [90] sont deux techniques de sélection qui rassemblent sélection de commandes et action au sein d'un même geste. Basés sur le principe du franchissement [1, 4], les *Control Menus* passent de l'étape sélection à l'étape action lorsque l'utilisateur franchit une certaine distance avec le premier point de contact. L'inconvénient est que cette distance est invisible. Il est donc difficile d'anticiper précisément le moment et l'endroit où va commencer le contrôle continu. Avec les *FlowMenus*, la délimitation est marquée par un retour sur le premier point de contact : ceci permet de maîtriser plus facilement le basculement vers le contrôle continu. Toutefois, le geste est complexe et le temps d'exécution plus long que les *Control Menus* [88].

Les délimiteurs spatiaux ne sont pas limités aux boutons virtuels. Sur ordinateur standard, l'utilisation de touches spéciales au clavier appelées *modifier keys*

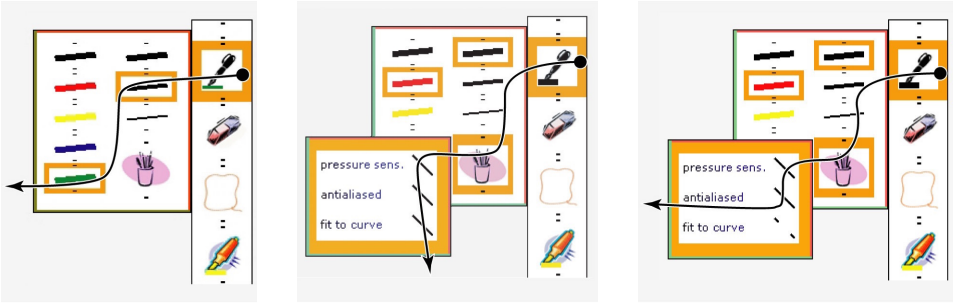


Figure 3.9 – *CrossY* [4]. Pour actionner les interacteurs graphiques, il suffit de les traverser [1, 4, 5].

(« *control* », « *alt* », « majuscule », « *command* ») permet de délimiter des quasi-modes [198] notamment avec la main non-préférée [186]. La touche « verrouillage majuscule » marque elle le basculement entre deux modes d'interprétation des touches alphanumériques. Bien qu'un clavier physique soit rarement disponible avec une tablette tactile, la littérature a souvent exploré les touches et les boutons comme délimiteurs, en complément d'une surface tactile [39, 89, 90, 100, 134, 157, 198, 216, 240]. Sur ordinateur standard, Doucette et al. ont utilisé un quasimode avec une touche spéciale pour accélérer le pointage sur une barre d'outils [67]. Il proposèrent trois techniques : (1) téléporter la barre d'outils pour la faire coïncider avec la position du curseur, (2) téléporter le curseur pour le faire coïncider avec la barre d'outils puis le renvoyer à sa position d'origine une fois la sélection effectuée, et (3) utiliser un écran tactile manipulé en temps normal comme dispositif de pointage indirect et s'en servir pour afficher la barre d'outils. S'agissant des tablettes, on pourrait s'inspirer de ces méthodes pour afficher la barre d'outil sous le doigt de l'utilisateur (ou à la dernière position enregistrée).

Les boutons disponibles sur certains stylets ont également été utilisés pour la navigation entre modes [47, 89, 141, 216].

Délimiteurs temporels

Un délimiteur particulièrement simple est l'écoulement du temps [99, 115, 141, 240]. Par exemple, un délai est un délimiteur qui se déclenche si une certaine action, comme le déplacement d'un point de contact sur la surface, n'est pas survenue avant une échéance temporelle donnée. Sur les interfaces tactiles, un bref délai sert souvent à ouvrir un menu contextuel. Avec les *Marking Menu*, ce délai sert, comme on l'a vu, à déclencher le mode novice [126]. Il est à nouveau mis en œuvre au sein du mode novice pour passer d'un niveau de menu à un autre [127].

Par définition, les délais ralentissent malheureusement l'interaction et ils sont donc peu adaptés pour les utilisateurs experts [141, 215].

Hinckley et al. ont estimé qu'un délai d'activation de 500 ms était à la fois suffisant pour minimiser les activations accidentelles et acceptable pour l'utilisateur [99]. Pour la segmentation, la durée du délai dépend des mouvements à délimiter. Kurtenbach a utilisé un délai de 150 ms pour passer en mode novice dans les *Marking Menus*.

La dimension temporelle est souvent combinée avec d'autres dimensions. Par exemple, un délimiteur utilisant un critère de vitesse ou un tap avec une durée maximale sont des délimiteurs gestuels temporels.

Délimiteurs gestuels mono-contact

Les délimiteurs gestuels sont des délimiteurs qui exploitent la forme du geste. Si l'on s'intéresse à des gestes effectués à la surface d'une tablette tactile, on analysera l'évolution du contact. Les délimiteurs gestuels qui ne recouvrent aucune autre catégorie ne prennent en compte ni le temps (délimiteurs temporels) ni les coordonnées du contact (délimiteurs spatiaux). Seuls sont pris en considération leurs apparitions (lorsque le contact est établi), leurs disparitions (lorsque le contact est interrompu) et leurs déplacements.

Sur tablette, le délimiteur gestuel le plus simple est le tap qui correspond à la séquence apparition-disparition sans déplacement [240]. Ce délimiteur est souvent associé à la dimension spatiale et est utilisé lors de la plupart des tâches de pointage, notamment pour l'activation de bouton. Il correspond à l'équivalent d'un clic de souris. Le double tap (correspondant à la séquence apparition-disparition-apparition-disparition sans déplacement) a également été mis en œuvre [64], par exemple pour zoomer sur une carte dans la plupart des applications de cartographie commerciales.

Lorsque la principale source d'entrée n'est pas l'écran *multitouch*, se reposer uniquement sur le contact avec la surface peut suffire. Par exemple, Hinckley et Song ont proposé d'interpréter les mouvements d'un téléphone lorsqu'on maintient un doigt appuyé sur la surface de son écran [103].

Une des études les plus connues dans la littérature IHM consacrée au problème des délimiteurs est celle de Hinckley et al. sur les *pigtails* [99]. Il s'agit de petites boucles, c'est à dire un trait qui se recroise lui-même (Figure 3.2). Hinckley et al. ont imaginé s'en servir pour l'ouverture d'un menu contextuel après avoir entouré les objets d'intérêts. Roudaut et al. et Bonnet et al. sont capables de reconnaître la signature d'un roulement de doigt sur la surface [38, 183]. Olwal et al. détecte les mouvements oscillatoires linéaires un peu comme si l'on grattait l'écran [166]. Sur papier augmenté, Tsandilas et Mackay distinguent les « nœuds » — c'est-à-dire de gros points — de l'écriture cursive [214].

Malacria et al. détecte les mouvements elliptiques et sélectionne différents modes de contrôle en fonction du rapport entre les rayons de l'ellipse [144]. Enfin, Appert et Bau ont développé un algorithme capable de détecter l'échelle d'un geste [6], ce qui peut également faire office de délimiteur.

Certains délimiteurs gestuels exploitent également la dimension temporelle afin, par exemple, de tirer parti de la cinétique du mouvement. Ainsi la durée maximale des taps et des double taps est souvent limitée. Moyle et Cockburn ont proposé d'utiliser des « *flicks* » pour naviguer dans de longues listes de lignes de texte, comme l'historique d'un navigateur internet [156]. Ces *flicks* doivent être tracés suffisamment rapidement pour être reconnus.

Il est souvent difficile d'utiliser un délimiteur gestuel mono-contact pour l'activation car les actions interprétées par un mode sur tablette tactile repose souvent également sur des gestes mono-contact (par exemple pour déplacer une image). Il est donc difficile de s'en différencier. Pour pallier cette difficulté plusieurs méthodes peuvent être utilisées :

— Exploiter les mouvements sans effet dans le mode par défaut (par exemple un tap). Il s'agit de la méthode préférentielle mais elle n'est pas toujours possible car certains modes par défaut utilisent n'importe quel mouvement (par exemple un mode dessin).

— Commencer à interpréter les mouvements dans le mode par défaut puis les annuler dès que le délimiteur est reconnu. Zeleznik et Miller utilisèrent une méthode similaire : par défaut le mouvement est interprété comme du dessin puis, si le délimiteur d'une commande gestuelle est reconnu, ce dessin est effacé et le geste est utilisé pour exécuter une commande. Ceci provoque toutefois des manipulations incidentes à travers le mode source ce qui peut être désagréable.

— Retarder l'interprétation du mode par défaut jusqu'au moment où la reconnaissance du délimiteur est invalidée (ou validée).

Délimiteurs gestuels *multitouch*

L'utilisation du *multitouch* ouvre la voie à la reconnaissance de gestes plus complexes et qui se distinguent mieux des modes par défaut utilisés dans les applications. Par exemple, Bailly et al. et Damaraju et al. ont proposé d'interpréter différemment les mouvements des doigts selon le nombre des contacts [21, 64]. Similairement, Au et Tai ont conçu un menu qui s'ouvre quand on tape la tablette avec quatre doigts ou plus [10]. Le critère de la posture manuelle a été également plusieurs fois utilisés [94, 139, 149], par exemple pour mimer la manière dont est manipulé un objet physique (Figure 3.10) [94].

Le *Multi-Finger Pie Menu* de Banovic et al. (Figure 3.11) est un menu circulaire qui est activé dès que l'utilisateur touche la surface avec le doigt [24]. Toutefois, l'affichage du menu n'apparaît que progressivement et s'efface au moindre mouvement ce qui lui permet de ne pas interférer avec les interactions par défaut. Les éléments du menu sont sélectionnés avec un autre doigt de la main. Ce menu est particulièrement intéressant, car il n'a pas besoin d'être explicitement activé et pour autant il ne déborde pas sur les interactions par défaut.

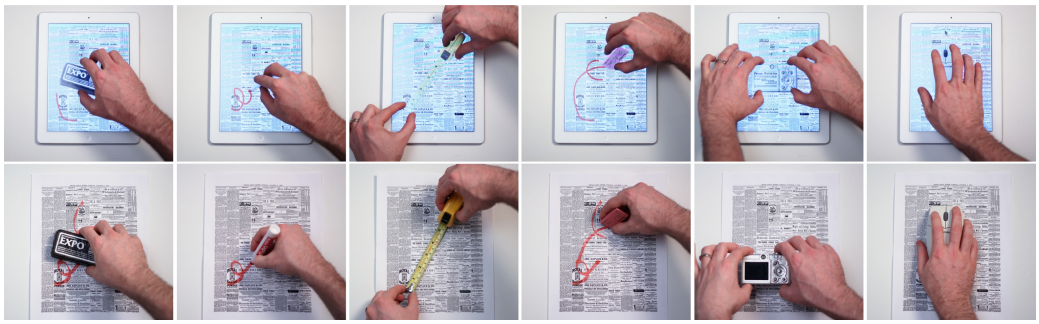


Figure 3.10 – TouchTools [94]. Le système change de mode en fonction de la posture de la main. Les postures évoquent l'utilisation d'objets réels.

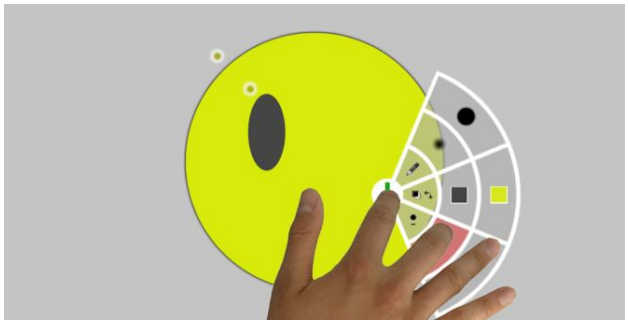


Figure 3.11 – *Multi-Finger Pie Menu* [24]

Orientation, son et image, localisation

Une tablette tactile n'est pas uniquement équipée d'une surface tactile. L'accéléromètre, le micro, la caméra, le *bluetooth* peuvent également offrir des dimensions intéressantes pour la conception de délimiteurs. À ces périphériques, on peut également ajouter le clavier.

Orientation et accéléromètre. De nombreux travaux se sont intéressés à l'accéléromètre comme délimiteur [12, 103, 180, 187, 199, 213, 216]. Par exemple, des techniques tirant parti de l'orientation de l'appareil (*tilt techniques*) [172] permettent l'alternance entre les différents modes disponibles au sein d'une application [213] ou le passage d'une application à une autre (par exemple *TimeTilt* [180]). Des mouvements de va-et-vient rapides, appelés « *jerks* » peuvent aussi être exploités [12, 180, 187, 216] (par exemple *JerkTilts* [12] ou *DoubleFlip* [187]). Enfin, l'accéléromètre offre la possibilité de détecter des taps en dehors de la surface tactile, par exemple en marge de l'écran comme avec *Bezel-Tap* [199] ou dans le dos de l'appareil [180].

Spindler et al. ont décrit une technique, utilisable sur les tablettes et sur les téléphones, qui permet de naviguer dans une scène en fonction des mouvements de

l'appareil dans l'espace 3D [204]. Cette navigation était activée lorsqu'un contact était détecté sur la surface tactile (par défaut rien ne se passait). L'entrée sur la surface tactile servait donc de délimiteur pour un quasimode où les mouvements dans l'espace sont interprété différemment.

Son. La reconnaissance de la parole est utilisée par Apple et Google pour lancer la reconnaissance vocale de commandes. Le délimiteur repose sur une petite phrase clé : « *Hey Siri* » pour Apple et « *OK Google* » pour son concurrent. Ce type d'interface n'est pas nouveau et a déjà fait le fruit de plusieurs travaux dans la littérature, notamment dans le domaine de la réalité augmentée ou des interfaces multimodales [37, 62]. L'analyse sonore en continu est toutefois souvent énergivore, ce qui pose problème pour les usages mobiles.

Image vidéo. À ma connaissance, la caméra vidéo a rarement été utilisée comme canal d'interaction avec une tablette. Toutefois, si la tablette est à plat, on pourrait imaginer reconnaître un « *pinch* » en l'air (Figure 3.12) [231] pour, par exemple, zoomer et dézoomer avec un mouvement normal à la surface. Malheureusement, de même que l'analyse sonore, l'analyse vidéo consomme beaucoup d'énergie, ce qui compromet sa mise en œuvre sur les dispositifs mobiles.

Localisation de l'utilisateur. L'utilisation de la localisation comme délimiteur est limitée à des cas assez particuliers. Par exemple, le jeu mobile *Ingres* débloque certaines actions en fonction de la localisation de l'utilisateur. En se servant du *Bluetooth*, on pourrait imaginer sortir automatiquement du mode verrouillé lors de l'activation d'une tablette ou d'un mobile si un autre appareil, connecté au même compte utilisateur, est déverrouillé et à proximité.

Les délimiteurs sur les autres plateformes

Si la surface tactile est capable de reconnaître précisément la forme du contact, cette forme peut être utilisée pour marquer un délimiteur [15, 49, 76, 238]. Par exemple, Bailly et al. ont proposé un menu s'ouvrant lorsqu'on repose le talon de la main sur la surface [15]. La technologie majoritairement utilisée sur les tablettes tactiles, qui renvoient peu d'information sur la forme des contacts avec la surface rend toutefois ce type de délimiteur encore difficile à mettre en place.

Les délimiteurs sont un problème critique pour l'interaction gestuelle en l'air où l'on doit pouvoir distinguer les mouvements naturels de l'utilisateur des mouvements adressés au système [27]. Un des délimiteurs en l'air les plus connus est le *pinch* qui consiste à toucher son pouce avec son index (voir la Figure 3.12) [231]. Similairement, Walter et al. ont travaillé dans le domaine des écrans publics interactifs et ont proposé de poser la main sur la hanche [224].

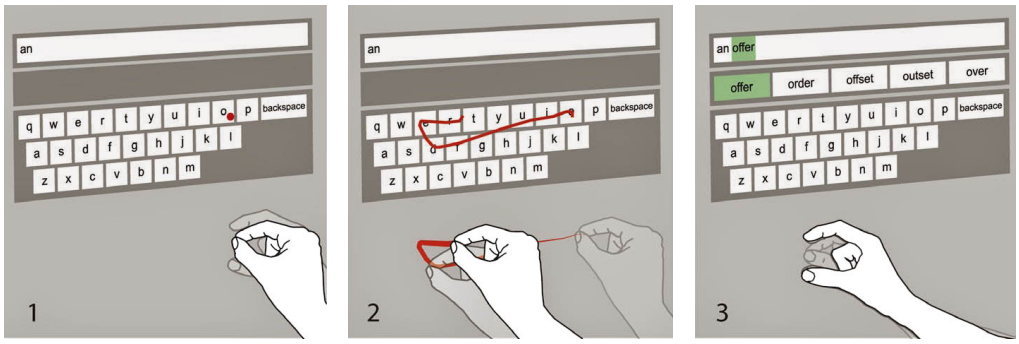


Figure 3.12 – Utilisation du *Pinch* en l'air [231] pour l'entrée de texte [146].

Parmi les autres types de canaux d'entrée mis à profit dans le cadre des délimiteurs on peut compter :

— La pression [141, 173, 216],

— Le survol [71, 72, 86] et enfin

— L'identification des doigts (voir le Chapitre 6) [32, 139, 147].

3.3 Conclusion du chapitre

L'utilisation des modes, à l'exception des modes spaciaux, est en général déconseillée et les guides de conceptions prônèrent jusque dans les années 90 les interfaces dites « *modeless* »[8, 209]. En réalité, la définition donnée au concept de mode est longtemps restée floue. En cherchant à le définir précisément, nous nous apercevons que les modes sont omniprésents et inévitables : rien qu'un bouton sur une interface graphique est une manière de changer l'interprétation du clic de la souris lorsqu'il est survolé par le curseur. Le défi réel lors de la conception d'une application n'est donc pas de minimiser autant que possible l'usage des modes, mais d'en maîtriser soigneusement les transitions, c'est-à-dire de définir des délimiteurs efficaces et appropriés.

Sur tablette, l'utilisation de modes spaciaux pose problème car l'espace-écran est précieux et doit être consacrée autant que possible aux objets d'intérêt. La définition d'un délimiteur de mode non spatial est difficile, car celui-ci doit s'accommoder du mode courant et des interactions qu'il permet. De plus, l'étendue du champ des possibles n'en permet pas une exploration méthodique puisqu'il suffit de « quelque chose de différent » [101]. Toutefois l'importance des délimiteurs, et en particulier le délimiteur d'activation pour une technique de menu, est capitale. La plupart des techniques d'interaction de la littérature ont été proposées dans le contexte d'une interaction à la souris avec laquelle les boutons font facilement office de délimiteur. Ainsi, elles sont dotées d'un délimiteur d'activation peu efficace pour se distinguer

au sein d'un mode de contrôle continu sur tablette comme le déplacement d'une image par exemple : souvent la technique est activée dès que l'utilisateur pose le doigt sur l'écran [17, 125, 126, 169].

Il peut être intéressant d'explorer une approche inverse et de réfléchir davantage à l'intégration du délimiteur d'activation au sein de la technique, en particulier dans le cas des applications sur tablettes tactiles. De plus, il devient alors possible d'exploiter les degrés de liberté non utilisés de ce délimiteur au sein de la technique.

4 Techniques de changement de modes

Suite aux conclusions du Chapitre 3, nous avons entrepris de développer plusieurs techniques d'interaction permettant d'alterner entre différents modes de contrôle en respectant les principes suivants :

1. Intégration d'un délimiteur d'activation au sein de la technique d'interaction. Notre analyse des délimiteurs (Chapitre 3) souligne l'importance et le potentiel de cette propriété peu exploitée dans la littérature. Elle permet à une technique de sélection de s'accommoder facilement de son mode source – c'est-à-dire le mode à partir duquel elle est activée – tout en permettant l'exploitation des degrés de libertés éventuels du délimiteur mis en place.

2. Pas de consommation d'espace-écran tant que la technique n'est pas activée. L'objectif est de garder la totalité de la surface d'affichage disponible pour la tâche utilisateur.

3. Raccourcis experts et transition de novice à expert par révélation, guidage et répétition suivant les principes instaurés par Kurtenbach et al. [59, 128, 132] (voire la Partie 2.3.5). Cette propriété améliore grandement la vitesse à laquelle un utilisateur gagne en performances avec la technique.

Additionnellement, nous intéressons à la fusion de la sélection et du contrôle continu au sein du même mouvement. Les travaux de Pook et al. [169] et Guimbretière et al. [88, 90] ont montré que ceci pouvait permettre une amélioration significative des performances.

Nous avons conçu deux techniques d'interaction, chacune utilisant un délimiteur d'activation différent. La première, le *FingerMenu*, intègre un délimiteur d'activation gestuel *multitouch* (Partie 4.1). La seconde, le *SigmaMenu*, intègre un délimiteur d'activation gestuel mono-contact (Partie 4.2).

Nous avons évalué le *SigmaMenu* et le *FingerMenu* à l'aide de deux études utilisateurs où les participants devaient alterner entre différents modes de contrôles. Dans le but d'être le plus proche possible d'un contexte et d'une tâche réels, la première a été exécutée directement au sein du prototype de *GE HealthCare* (Partie 4.3). La seconde a été plus formelle et a été exécutée en dehors du prototype (Partie 4.4).

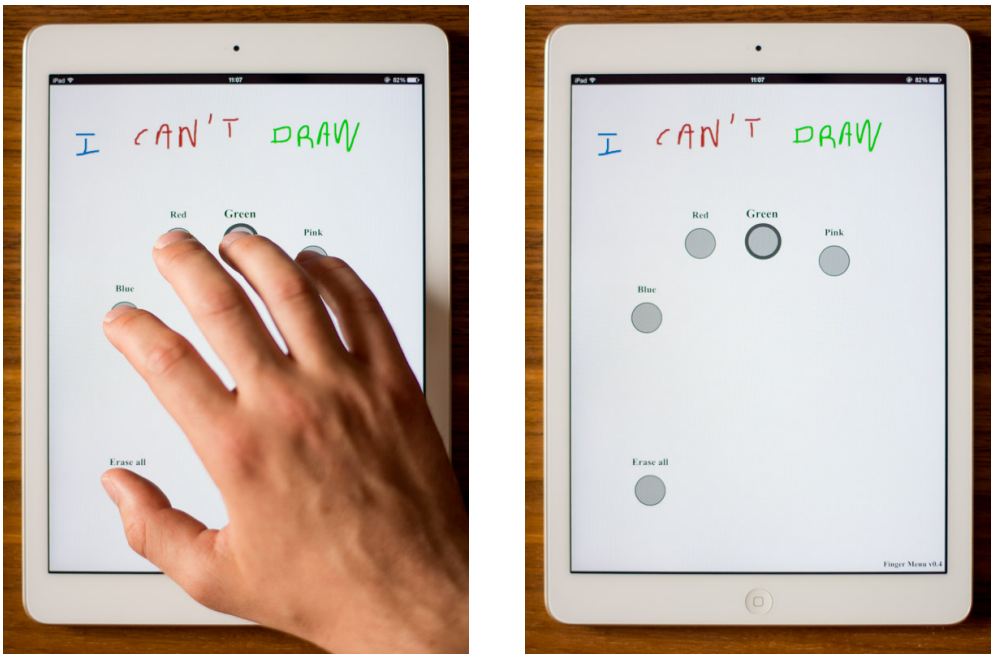


Figure 4.1 – Activation d'un *FingerMenu*

4.1 FingerMenu

Le *FingerMenu* repose sur un délimiteur d'activation gestuel *multitouch*. Similairement à Au et Tai [10], la technique est activée en posant quatre doigts ou plus sur la surface (voir la Figure 4.1). Le menu est alors affiché sous la forme de « pastilles » apparaissant sous chacun des doigts. Chaque commande est associée à une pastille qui peut être activée à la manière d'un bouton (voir la Partie 4.1.1). Ceci permet une association commande-doigt : chaque doigt est lié à une commande différente. Notre hypothèse est que cette propriété peut permettre de faciliter la mémorisation des commandes et ce pour plusieurs raisons :

1. Tout d'abord ceci fait appel à un type de mémorisation assez peu exploité lors de l'interaction avec une interface graphique : la mémorisation de différentes partie du corps,
2. En second lieu, utiliser un doigt en particulier lors de la manipulation donne un retour proprioceptif sur ce doigt : l'utilisateur a conscience, tout au long de la manipulation, du doigt qui est engagé. Traditionnellement, le doigt n'a pas d'importance et cette information n'est donc pas pertinente pour l'utilisateur. Dans le cas du *FingerMenu*, toutefois, le retour proprioceptif rappelle à l'utilisateur le mode activé ainsi que la manière dont le sélectionner tout au long de la manipulation.

Il est aussi possible de différencier la main gauche de la main droite à condition de poser les cinq doigts (l'identification de la main se basant sur celle du pouce).

Afin d'éviter un retour utilisateur intempestif pouvant déranger un expert déjà familier avec les différentes commandes, le menu apparaît avec un léger retard. Cette méthode est inspirée des *Marking Menu* [128, 132]. Elle contribue à mettre en place une transition de novice à expert par révélation, guidage et répétition suivant les principes également instaurés par KURTENBACH et al. [59, 128, 132].

Le *FingerMenu* a été implémenté en HTML et JavaScript. Une démonstration en ligne est accessible à l'adresse `http://quentinroy.fr/demos/cube?menu=fingermenu`. Elle a été testée sur Safari Mobile 8.0. Cette technique nécessite bien sûr que l'écran soit *multitouch*.

4.1.1 Syntaxes d'interaction

Nous avons défini plusieurs syntaxes pour interagir avec le *FingerMenu*. Elles sont toutes compatibles et l'utilisation de l'une ou de l'autre peut donc être laissée aux choix de l'utilisateur final. Leur compatibilité est garantie par l'utilisation de délimiteurs légèrement différents.

La machine à état représenté par la Figure 4.2 montre le fonctionnement du *FingerMenu* et de ces différentes syntaxes.

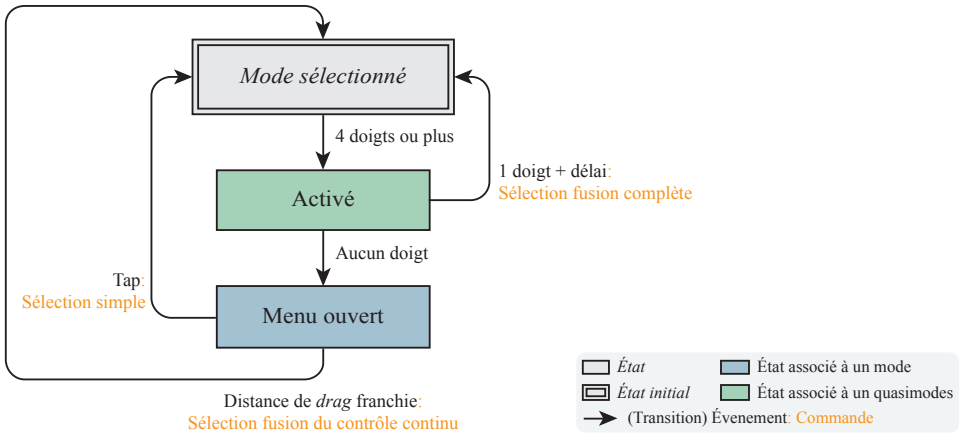


Figure 4.2 – Machine à état simplifiée du *FingerMenu*

Sélection simple

1. L'utilisateur tape au moins avec les quatre doigts longs de la main. Le menu apparaît.
2. L'utilisateur tape sur la pastille correspondant à la commande cible.

En pratique, l'utilisateur peut utiliser n'importe quel doigt pour toucher une pastille. Toutefois, juste après l'activation, le doigt associé est positionné juste au-dessus d'elle. Il est donc particulièrement prédisposé à cette tâche : en utiliser un autre implique une tâche de pointage sensiblement plus difficile. L'apparition des pastilles à l'emplacement de chaque doigt tend à encourager ce comportement. Notre hypothèse est qu'un utilisateur l'adoptera rapidement pour améliorer sa vitesse de sélection.

Cette syntaxe est la plus simple et permet facilement de découvrir les différentes fonctionnalités. Il s'agira sans doute de la syntaxe utilisée par un utilisateur lors de la découverte de la technique.

Fusion du contrôle continu

Lorsque la commande sélectionnée sert à activer un mode permettant du contrôle continu, le *FingerMenu* permet de fusionner la sélection de ce mode avec le contrôle continu au sein d'un seul et même mouvement en surface, sans relever le doigt.

1. L'utilisateur tape au moins avec les quatre doigts longs de la main. Le menu apparaît.
2. L'utilisateur repose un doigt sur la pastille correspondant à la commande cible sans le relever. Là encore, rien ne l'empêche en pratique d'utiliser un autre doigt que celui associé à la pastille. La remarque ci-dessus reste donc valide.
3. Il commence à déplacer ce doigt sur la surface. Si la commande sélectionnée est associée à l'activation d'un mode, celui-ci est enclenché et sa manipulation commence immédiatement.

Un tap impliquant souvent quelques mouvements du point de contact avant que le doigt ne soit relevé, un léger seuil de déplacement est utilisé afin de différencier cette syntaxe de la précédente

Cette syntaxe permet d'accélérer la transition entre la sélection de la commande et la manipulation continue.

Fusion complète

Cette dernière syntaxe fusionne le délimiteur d'activation, la sélection de commande et le contrôle continu au sein d'un seul et même mouvement en surface.

1. L'utilisateur pose les cinq doigts de la main sans les relever. Si l'utilisateur tarde à passer à l'étape 2, le menu apparaît.
2. L'utilisateur relève tous les doigts sauf celui correspondant à la commande cible.

3. L'utilisateur commence à déplacer le doigt restant (nécessairement celui associé à la pastille) sur la surface. Si la commande sélectionnée correspond à l'activation d'un mode, celui-ci est enclenché et sa manipulation commence immédiatement.

Les doigts n'étant jamais exactement relevés au même moment, un léger délai est utilisé afin de différencier cette syntaxe des deux précédentes.

Cette syntaxe est plus avancée et est destinée aux utilisateurs experts. Elle permet d'accélérer l'ensemble des transitions entre modes depuis l'activation de la technique jusqu'à la manipulation continue.

On remarquera que ces trois syntaxes ne diffèrent que très légèrement et que l'apprentissage des commandes avec l'une est susceptible de profiter tout autant aux deux autres.

4.1.2 Mise en place de modes transitoires

Si une commande est associée à l'activation d'un mode de contrôle continu, il est possible de tirer parti des différentes stratégies d'utilisation du *FingerMenu* pour permettre à l'utilisateur de décider s'il souhaite manipuler ce contrôle de manière « permanente » (le mode sélectionné devient le mode par défaut) ou transitoire. Un mode transitoire ne sera actif que pour une et une seule opération. Après sa désactivation, typiquement (mais pas nécessairement) délimitée par le moment où l'utilisateur relâche son doigt de la surface, le système re-bascule sur le mode par défaut.

Un mode transitoire est utile dans le cas d'actions ponctuelles, exécutées une fois de temps en temps. Par exemple, l'utilisateur d'une application de dessin à main levée utilise souvent majoritairement l'outil crayon, mais peut périodiquement recourir à l'outil gomme pour effacer un trait mal placé ou à l'outil « étaler » pour atténuer le tracé à une position donnée.

Nous proposons de recourir à la méthode de sélection simple sans fusion pour remplacer le mode par défaut et d'utiliser les deux autres méthodes (fusionnant la sélection et le contrôle continu) pour activer un mode transitoire. La Figure 4.3 représente le *FingerMenu* équipé de ce fonctionnement.

4.1.3 Algorithme d'identification des doigts

Au et Tai [10] proposent un algorithme reposant sur les angles entre les segments joignant le barycentre des points de contact de chacun des doigts. Ces angles sont appelés « *spanning angles* » (voir la Figure 4.4). Le pouce est identifié comme le point dont la somme des angles est la plus importante. L'index est ensuite déterminé comme le point le plus proche du pouce, le majeur comme le doigt le plus proche de l'index à l'exception du pouce et ainsi de suite. Ewerling et al. [69] ont conçu

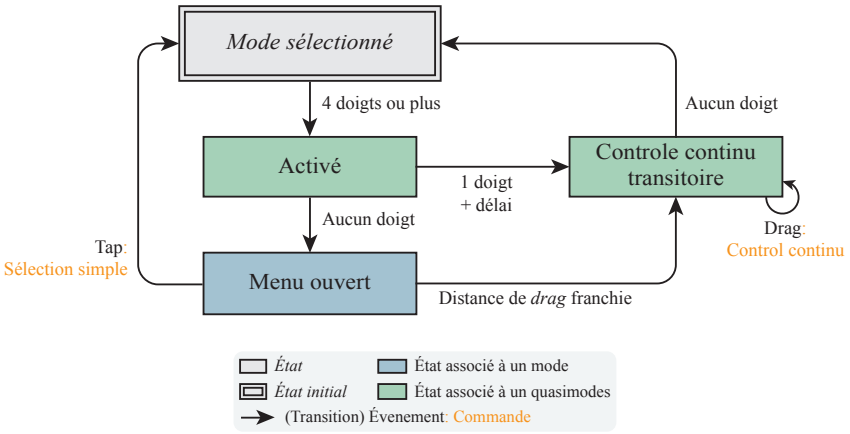


Figure 4.3 – Machine à état simplifiée du *FingerMenu* avec modes transitoires

un algorithme légèrement différent qui ordonne les doigts en utilisant un algorithme du plus court chemin. Le pouce est alors identifié comme le point le plus éloigné du barycentre de l'ensemble des autres points de contact, et les doigts sont identifiés dans le sens de l'ordonnancement. L'inconvénient de ces deux méthodes est que le pouce doit être présent.

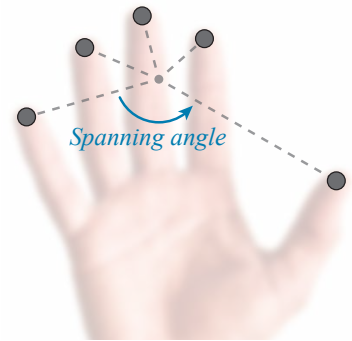
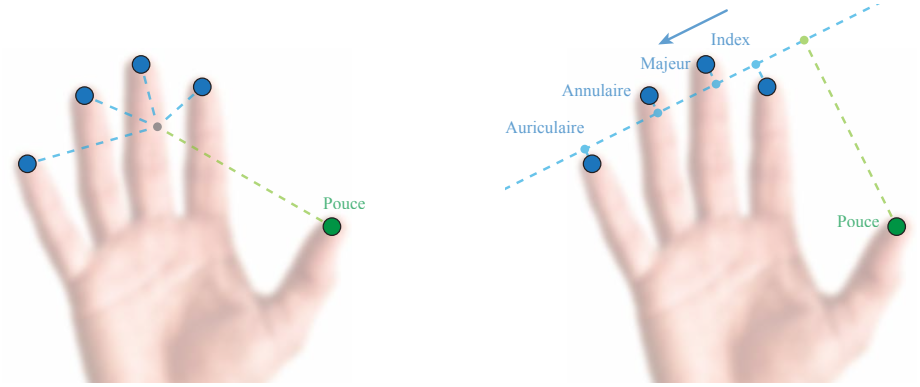


Figure 4.4 – *Spanning angle* entre points de contact au sein de l'algorithme de reconnaissance des doigts de Au et Tai [10]

L'algorithme de reconnaissance du *FingerMenu* commence par identifier le pouce comme le point le plus éloigné du barycentre des point de contacts (Figure 4.5a). Si seulement 4 points de contacts sont détectés, cette étape est ignorée (on considère dans ce cas que le pouce n'est pas en contact). L'algorithme s'appuie ensuite

sur une régression linéaire orthogonale [1] de la position des doigts longs pour les ordonner (Figure 4.5b). Si le pouce est présent, sa position indique le sens de l'identification, et donc la main (Figure 4.5b). Sinon, le sens dépend de la main par défaut (typiquement la main droite).



(a) S'il y a 5 points de contact, identification du pouce comme le point de contact le plus éloigné du barycentre.

(b) Calcul de la régression linéaire orthogonale de la position des doigts longs. Identification des doigts dans l'ordre des projections sur cette droite.

Figure 4.5 – Algorithme d'identification des doigts à partir de cinq points de contact

L'avantage de cette méthode est que les doigts peuvent être identifiés même si le pouce n'est pas en contact. Toutefois, elle échoue avec toute autre combinaisons de 4 doigts (par exemple pouce, index, majeur et annulaire). Malgré cette limitation potentiellement source d'erreurs pour un utilisateur novice, nos tests informels indiquent que l'activation à seulement 4 doigts peut permettre de gagner en rapidité et nous avons donc décidé de l'implémenter.

À noter que l'utilisation d'une régression linéaire orthogonale (méthode des moindres carrés totaux) améliore sensiblement le taux de reconnaissance (nous n'avons constaté aucune erreur de reconnaissance avec cette méthode). Une régression linéaire verticale (la plus commune) calcule uniquement les distances parallèles à l'axe Y ce qui peut mener à une identification des doigts erronée lorsque l'on touche la tablette de côté et que les points de contacts sont à peu près verticalement alignés (il s'agit d'un cas rare mais existant). La Figure 4.6 montre la différence entre une régression linéaire verticale et un régression linéaire orthogonale.

1. Une régression linéaire orthogonale minimise la distance prise perpendiculairement à la droite modèle entre chaque point et la droite modèle. Elle s'oppose aux régressions linéaires simples (verticales ou horizontales) qui minimisent la distance prise parallèlement à l'axe des abscisses (horizontales) ou l'axe des ordonnées (verticales) entre chaque point et la droite modèle [2]. La Figure 4.6 montre la différence entre une régression linéaire verticale et un régression linéaire

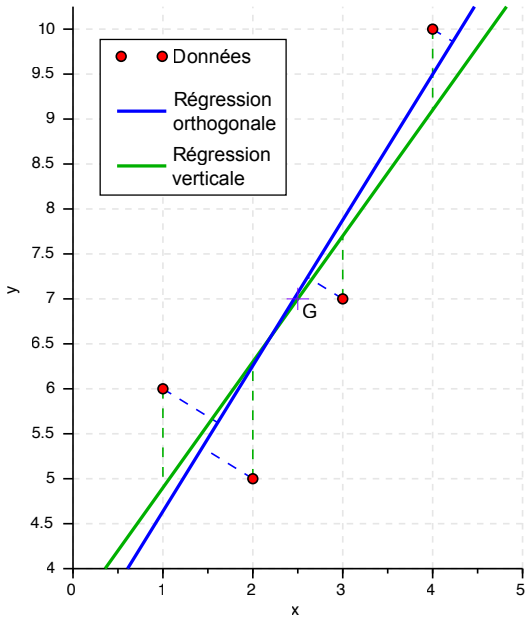


Figure 4.6 – Régression linaire verticale et régression linaire orthogonale

4.1.4 Multi niveaux

Plusieurs méthodes peuvent être mises en œuvre afin d'étendre le nombre de commandes accessibles avec le *FingerMenu*.

Rangées

Optionnellement, il est possible d'afficher plusieurs rangées de pastilles apparaissant les unes par dessus les autres (voir la Figure 4.7) à la manière du *Multitouch* Menu de Bailly et al. [15]. La position des pastilles supérieures est calculée afin de respecter l'orientation de la main. En fusion complète, seul le premier niveau est accessible. Toutefois, avec les autres stratégies, il est possible de sélectionner une des pastilles supérieures en décalant légèrement la main vers le haut avant de reposer le doigt.

Marking Menu sur le pouce

Une autre variante peut être de traiter le pouce comme un modifieur et d'y adjoindre un *Marking Menu* (un mécanisme à nouveau inspiré du à la manière du *Multitouch* Menu de Bailly et al. [15]). Une fois le menu activé en posant les

orthogonale.

2. https://fr.wikipedia.org/wiki/Régression_linéaire

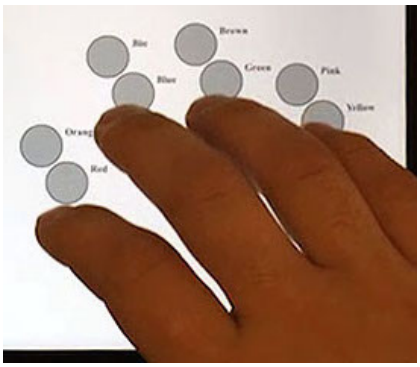


Figure 4.7 – Variante des FingerMenus avec plusieurs niveaux

cinq doigts de la main sur la surface et avant de les relever (mode « activé » sur la Figure 4.2 ou la Figure 4.3), déplacer le pouce sur la gauche, la droite, le haut ou le bas permet d'accéder a des sous-menus. La syntaxe d'interaction sans fusion devient donc :

1. L'utilisateur pose les cinq doigts de la main sans les relever.
2. Il déplace son pouce pour atteindre un des sous-menus.
3. Il relève les cinq doigts de la main.
4. Il tape sur la pastille correspondant à la commande cible pour la sélectionner.

Les deux autres syntaxes d'interaction présentées précédemment (fusion du contrôle continu et fusion complète, voir la partie 4.1.1) sont étendues de manière similaire.

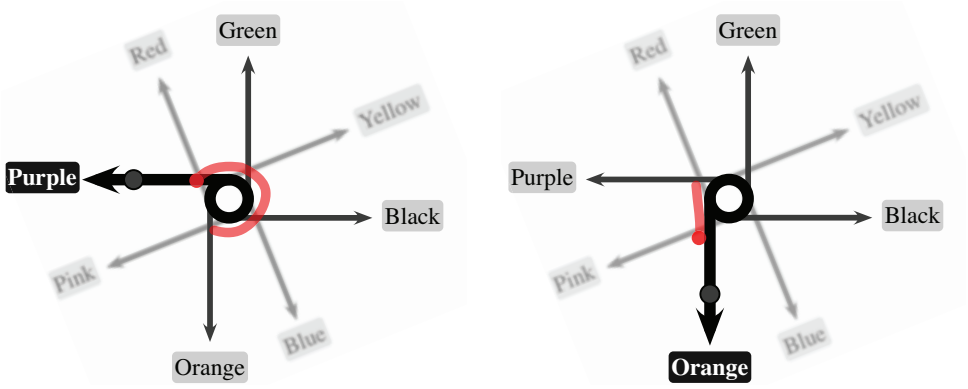
Il est alors possible d'atteindre jusqu'à (1 menu par défaut + 4 directions) × 5 doigts = 25 commandes différentes. Cette variante peut être combinée avec la précédente pour fournir davantage de commandes (50 avec deux rangées, 75 avec trois par exemple).

Marking Menu sur tous les doigts

La technique précédente adjoint au pouce un *Marking Menu* afin d'atteindre des sous-menus supplémentaires. Au lieu de ne prendre en compte que le déplacement du pouce pour déterminer la commande sélectionnée avec le *Marking Menu*, il est possible de prendre en compte le déplacement de l'ensemble de la main. Une fois le menu activé en posant quatre doigts ou plus sur la surface et avant de les relever (mode « activé » sur la Figure 4.2 ou la Figure 4.3), déplacer la main vers le haut, le bas, la gauche ou la droite permet d'accéder à des sous-menus. Cette variante à deux avantages sur la précédente : (1) il n'est pas nécessaire de poser le pouce pour accéder au sous-menu, (2) il est plus facile de déplacer l'ensemble de la main que seulement un doigt en gardant les autres fixes.

Cette méthode permet d'atteindre le même nombre de commandes que la précédente (25 commandes différentes) et peut également être combinée avec plusieurs rangées.

4.2 SigmaMenus



(a) Ouverture du menu (b) Préselection de « Orange »

Figure 4.8 – SigmaMenu en mode novice : sélection de la commande « Orange ». La trace rouge représente les mouvements de l'utilisateur et le point rouge la position courante du contact.

Le *Marking Menu* [126, 128] (abordé en 2.3.4) est un menu particulièrement efficace sur tablette tactile et est en général considéré comme la référence des menus gestuels. Notamment, il permet un apprentissage par la répétition et une transition fluide de novice à expert [59, 128, 132]. En effet, comme nous l'avons vu, la répétition de son mode novice permet à un utilisateur peu familier avec la technique d'apprendre peu à peu à utiliser le menu comme un expert : le mouvement du menu est le même dans les deux cas, seule la vitesse d'exécution (et l'apparition ou non du menu) change.

Toutefois, le *Marking Menu* ne permet d'atteindre qu'un nombre limité de commandes par niveau : 8 au maximum (12 si il n'y a qu'un seul niveau) [127]. De plus, ils n'intègrent pas de délimiteur permettant d'en distinguer l'activation d'un mode par défaut comme le déplacement du contenu dans une application affichant une carte ou des images.

Nous proposons le *SigmaMenu*, une nouvelle technique d'interaction qui combine un délimiteur d'activation gestuel mono-contact (voir la Partie 3.2.2) avec un *Marking Menu* : la technique est activée grâce à un mouvement circulaire de petite taille (voir la Figure 4.8). La sélection d'une commande dépend ensuite de la direction du mouvement comme pour un *Marking Menu*. De plus, en fonction

du sens de rotation avec lequel le menu a été activé, deux jeux de commandes différents sont présentés. Nous appelons « roue horaire » (respectivement « roue antihoraire ») l'ensemble des commandes accessibles par un mouvement circulaire dans le sens horaire (respectivement antihoraire). La Figure 4.8 montre la manière de sélectionner une commande avec un *SigmaMenu*. Pour un utilisateur novice, qui ne connait pas encore la position des différents éléments, il est possible de changer de sens de rotation une fois le menu ouvert pour basculer d'une « roue » à une autre (voir la Figure 4.9). Enfin, dans le cas des commandes servant à activer un mode de contrôle continu, une fois une certaine distance franchie la sélection est validée et le contrôle continu s'enclenche immédiatement à la manière d'un *Control Menu* [169]. La Figure 4.10 représente les différents modes mis en jeux lors de l'interaction avec le *SigmaMenu*.

Le *SigmaMenu* présente trois améliorations par rapport au *Marking Menu*.

1. Il intègre un délimiteur d'activation, la sélection est donc effectuée dans la continuité du délimiteur sans faire de pause et sans angle aigu.

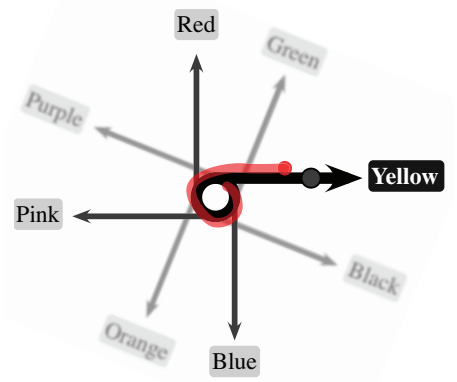
2. La taille maximale du vocabulaire de commandes est doublée grâce à l'exploitation du sens de rotation du délimiteur d'activation.

3. Il permet de fusionner la sélection de la commande au contrôle continu au sein du même mouvement [88, 90, 169].

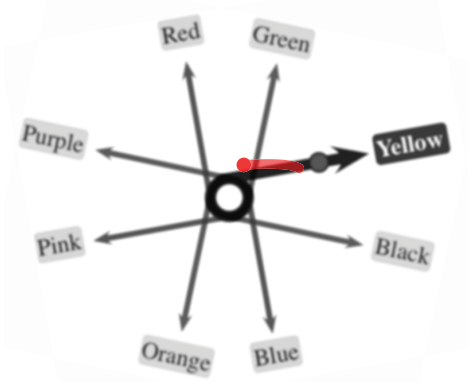
Le nom du *SigmaMenu* vient de la ressemblance entre la forme du geste sur la surface et la lettre grecque « σ ».

Le *SigmaMenu* a été implémenté en HTML, SVG et JavaScript. Une démonstration en ligne est accessible à l'adresse `http://quentinroy.fr/demos/cube?menu=sigmamenu`. Elle a été testée sur Safari Mobile 8.0, Safari 8.0, Mozilla Firefox 38.0 et Google Chrome 43.0.

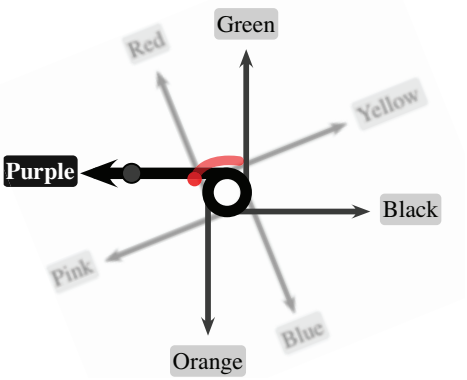
Dans cette partie, nous parlerons dans un premier temps de la mise en place des différents délimiteurs séparant les différents modes (et quasimodes) propres au *SigmaMenu* : d'une part le délimiteur d'activation (Partie 4.2.1), qui permet de séparer le mode à partir duquel est ouvert le *SigmaMenu*, et le mode de sélection de commandes, et d'autre part le délimiteur de sélection (Partie 4.2.2) qui, dans le cas où cette commande correspond à la sélection d'un mode de manipulation continu, permet de séparer la sélection de la manipulation de ce paramètre. Dans un second temps, nous discuterons de la mise en place du retour utilisateur (Partie 4.2.3) dont le rôle est de guider un utilisateur novice pour apprendre à se servir de la technique ainsi que de montrer les commandes disponibles. Nous terminerons en présentant quelques conceptions alternatives qui ont été considérées pendant la conception du menu (Partie 4.2.4).



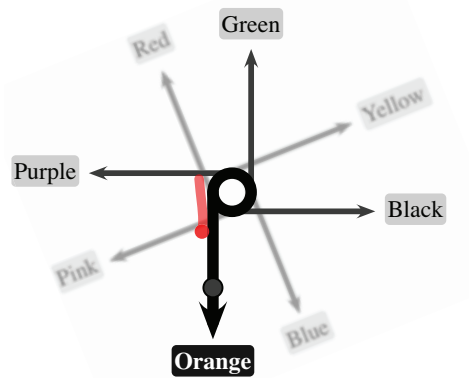
(a) Ouverture du menu en tournant dans le sens horaire. Le mode « Orange » est sur l'autre roue.



(b) Début du mouvement dans l'autre sens de rotation. Le menu pivote autour de son centre.



(c) Le menu a complètement pivoté. La roue horaire est passée « derrière ». La roue antihoraire est en place et activée.



(d) La commande « Orange » est présélectionnée.

Figure 4.9 – SigmaMenu en mode novice : sélection du mode « Orange ». La trace rouge représente les mouvements de l'utilisateur et le point rouge la position courante du contact.

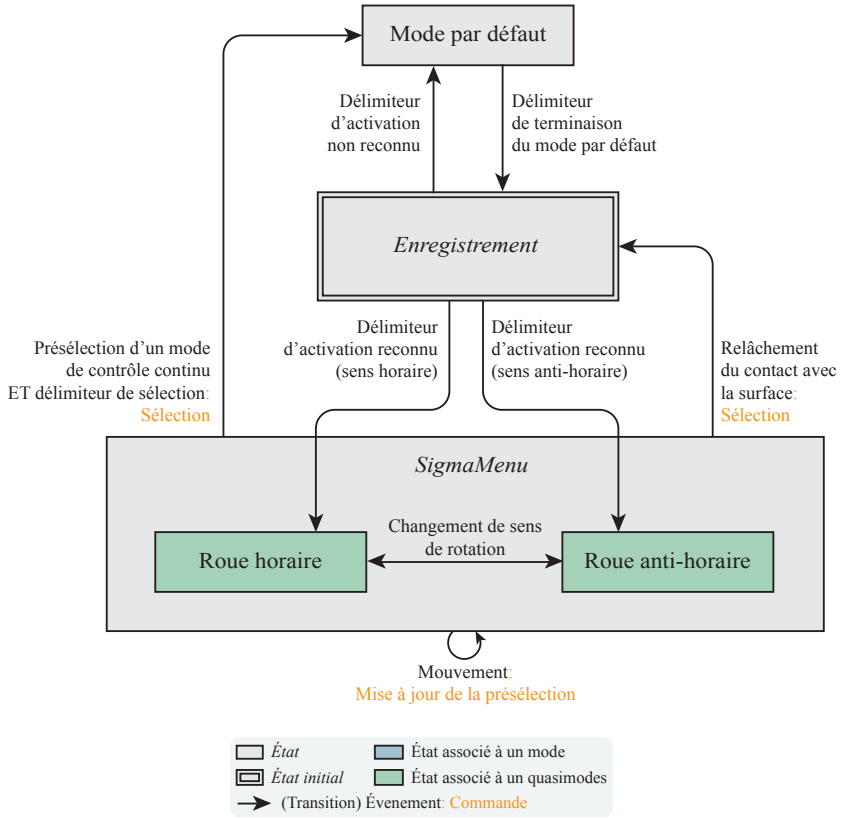


Figure 4.10 – Machine à états des différents modes impliqués lors de l'interaction avec un *SigmaMenu*

4.2.1 Mise en place du délimiteur d'activation

Le *SigmaMenu* est activé par un mouvement circulaire de petite amplitude. Ce délimiteur s'approche des *Knotty Gestures* de Tsandilas et Mackay [214] à l'exception que le sens de rotation est exploité et qu'il a été conçu pour une interface tactile. Sa forme ressemble également aux *pigtails* de Hinckley et al. [99] (Figure 4.11). Ceux-ci reposent cependant sur un croisement du tracé (« *self-crossing* ») et non pas sur la reconnaissance d'une forme circulaire. Une autre différence importante est que les *pigtails* servent à indiquer un changement de mode à la fin d'un tracé par exemple pour appliquer une commande sur un ensemble d'éléments sélectionnés grâce à un « lasso » comme dans *Bumptop* [2]. Le *SigmaMenu* n'a pas été initialement conçu pour cet usage, mais plutôt comme moyen de lancer une commande sans interférer avec le mode par défaut de l'application.

Figure 4.11 – *Pigtail* dans Scriboli [99]. La détection d'un croisement du tracé est utilisée pour ouvrir un *Marking Menu*.

Études préliminaires

Afin d'explorer l'utilisabilité d'un délimiteur d'activation circulaire, nous avons demandé à trente participants de tracer librement et sans retour visuel une succession de trois cercles sur une tablette tactile (un Apple iPad Air avec une diagonale d'écran de 22,8 cm).

Ce pilote a permis de montrer qu'il n'y avait pas de préférence forte sur le sens de rotation ce qui tend à démontrer l'utilisabilité de ce degré de liberté pour la sélection : en moyenne, sans instructions, les participants tracèrent 57,8 % de leurs cercles dans le sens horaire (σ = 43,8 %). Les participants ont toutefois eu une forte tendance à toujours conserver le même sens de rotation après avoir opté pour un en particulier (ce qui explique l'écart type important). Cela est sans doute dû au protocole expérimental qui demandait aux participants de tracer chaque cercle immédiatement à la suite.

Les participants tracèrent en moyenne des cercles d'un rayon de 2,01 cm (σ = 1,04 cm).

Lors d'un autre prétest, nous avons également demandé à trois participants

de tracer une série de cercles dont le sens de rotation était imposé. Ceux-ci ne reportèrent aucune difficulté particulière et le taux d'erreurs sans entraînement fut faible (94,4 % de succès, $\sigma = 5{,}60\,\%$).

Compétition avec le mode par défaut

Les mouvements circulaires sont rares lors des réglages continus par exemple l'ajustement de la luminosité, le déplacement ou l'agrandissement d'une image. Il s'agit donc d'un délimiteur d'activation intéressant pour une application où, par défaut, les gestes sur la surface activent ce type de fonctions.

La reconnaissance du délimiteur implique de capturer une partie du mouvement sur la tablette après un contact avant de pouvoir être validée (délimiteur reconnu) ou invalidée (délimiteur non reconnu). Cette étape, qui correspond à un mode à part entière (voir la Figure 4.10), est appelée « enregistrement » (*registration*) [76, 224, 229, 239].

Interpétation des mouvements durant l'enregistrement. Deux stratégies peuvent être mises en place pour interpréter les mouvements de l'utilisateur pendant l'enregistrement :

1. Si l'application comporte un mode de contrôle par défaut (comme déplacer une image lorsque l'on bouge le doigt sur la surface), effectuer ce mode par défaut pendant la phase d'enregistrement. C'est par exemple la stratégie utilisée par *Cyclostar* [144]. Il en résultera un petit déplacement du contenu qui peut éventuellement être annulé en revenant automatiquement à la position de départ lorsque le menu est activé. Cependant, cette solution n'est pas toujours applicable. En effet, il n'est pas toujours techniquement possible d'annuler les manipulations incidentes. De plus, celles-ci peuvent parfois s'avérer problématiques. Par exemple, dans le cadre de l'analyse radiologique, un radiologue cherche à atteindre précisément un artefact en particulier en se déplaçant longitudinalement dans une image 3D. Une fois ce point atteint, il est important de ne plus bouger sous peine de perdre l'élément à observer.
2. Ne rien faire tant que la reconnaissance du délimiteur n'a pas été validée ou invalidée. En conséquence, le mode par défaut parait retardée. Dans le cas de notre prototype de radiologie, nos tests informels semblent indiquer que cette dernière stratégie est peu dérangeante contrairement à ce que nous pensions au départ. Il s'agit de celle que nous avons mise en place lorsque nous avons intégré les *SigmaMenu* au prototype de radiologie de GE HealthCare et de celle représentée par la Figure 4.10.

Délimitation de l'enregistrement. L'enregistrement est délimité par la validation ou l'invalidation du délimiteur. Il est donc important de définir avec soin la manière

dont sont générés ces deux évènements. Selon la manière dont sont interprétés les mouvements pendant l'enregistrement, la priorité sera mise sur l'un ou l'autre :

1. Si l'enregistrement est confondu avec le mode source (stratégie 1 ci-dessus), il est préférable de mettre la priorité sur la validation du délimiteur afin de minimiser les interactions incidentes lorsque l'utilisateur souhaite ouvrir le menu.
2. Au contraire, si l'enregistrement attend l'invalidation du délimiteur avant de permettre au mode par défaut de prendre le relai (stratégie 2 ci-dessus), il est important que cette information soit donnée le plus rapidement possible afin de le retarder le moins possible.

Validation de la reconnaissance du délimiteur d'activation

Nous avons choisi d'activer la technique dès que 45 % d'un cercle est tracé. Le retour visuel est cependant affiché un peu plus tard : à peu près au 85 % du cercle ou après 750 ms. Ces valeurs ont été déterminées empiriquement à partir de nos tests avec la technique et ont pour objectif de permettre l'utilisation du menu en mode expert sans retour utilisateur. Ce dernier est inutile pour un expert et (comme tout menu) tend à détourner l'attention de l'utilisateur pour sa tâche. L'activation de la technique dès les 45 % permet de sélectionner une commande avec un bref arc de cercle. Le geste à alors la forme d'une canne (voir la figure 4.12).

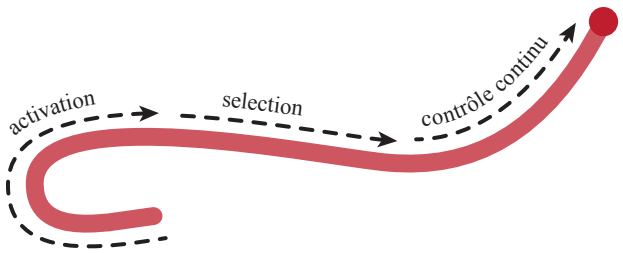


Figure 4.12 – Trace correspondant à la sélection et la manipulation d'un contrôle continu par un expert avec la technique. Le point rouge correspond à la position finale. L'activation a été validée, car au moins la moitié d'un cercle a été tracé au début du mouvement. Toutefois, le menu ne s'est pas affiché, car moins de 85% du cercle a été tracé.

Invalidation de la reconnaissance du délimiteur d'activation

Première méthode : frontière. La méthode la plus simple pour invalider la reconnaissance du délimiteur d'activation du *SigmaMenu* est tout simplement de mettre en place une distance de déplacement maximum après le contact avec la surface. Si, au-delà de cette frontière, un utilisateur n'a toujours pas activé

la technique (c'est-à-dire n'a toujours pas tracé au moins 45% d'un cercle), la reconnaissance est invalidée. Un corolaire de cette méthode est que cette distance maximum définit également le diamètre maximum du cercle pouvant être tracé (un cercle trop grand dépassera inévitablement la frontière). Notre étude préliminaire indique que les utilisateurs ont naturellement tendance à tracer en moyenne des cercles de 2,01 cm ($\sigma = 1{,}04$ cm). Si on souhaite respecter ce comportement, il est nécessaire de positionner la frontière à environ 4 cm du premier point de contact avant de pouvoir commencer à manipuler le mode par défaut dans le cas où rien ne se passe pendant l'enregistrement du délimiteur. Cependant, une distance aussi élevée pénaliserait l'interaction, car nécessiterait des gestes d'une amplitude importante pour lancer les commandes. Nous avons donc limité la taille du rayon à 120 pixels ce qui correspond à approximativement 1,14 cm sur un iPad Air. Empiriquement et une fois un utilisateur informé que le cercle doit être petit, nous avons constaté que cette taille était suffisante pour activer la technique d'une manière qui soit à la fois fiable et confortable tout en ne retardant pas trop le mode source.

Seconde méthode : courbure minimum. Afin de raccourcir la distance d'invalidation, il est possible de tirer parti de la détection de la courbure du tracé (celle-ci est de toute façon itérativement calculée par le reconnaisseur dans le but de reconnaître un cercle). L'idée est que si la courbure est trop faible, la reconnaissance peut être immédiatement invalidée. Toutefois, comme l'ont montré nos études préliminaires, les cercles tracés sont loin d'être parfaits. Nos pilotes avec le *SigmaMenu* montrent que, pour activer, les utilisateurs ont parfois tendance à démarrer leur mouvement de manière rectiligne avant de commencer à tourner (voir la Figure 4.12). Le dessin a alors la forme de canne ce qui est beaucoup plus rapide à tracer qu'un cercle bien fait. Si on souhaite autoriser ce comportement, il est nécessaire d'augmenter la quantité de mouvement à recueillir avant d'invalider la reconnaissance du délimiteur.

La détection de la courbure a toutefois le potentiel de réduire la durée de l'invalidation ainsi que d'augmenter la taille du plus grand cercle d'activation possible. Dans notre prototype, nous avons utilisé la première méthode, davantage de tests utilisateur restent nécessaires pour évaluer la seconde et optimiser ses performances.

Reconnaisseur

Le reconnaisseur doit être capable d'estimer le rayon et le centre du cercle passant au plus près des points de contact enregistrés.

Notre reconnaisseur s'appuie sur la méthode géométrique de la moyenne des intersections[3] [217]. Son algorithme calcule, pour chaque combinaison de trois points parmi les points du tracé, le cercle circonscrit à ces trois points. La moyenne

3. https://fr.wikipedia.org/wiki/Régression_circulaire

de ces cercles donne une estimation du cercle passant au plus proche de tous les points. Son rayon correspond à la courbure du segment.

La reconnaissance d'un cercle est itérativement appliquée sur les mouvements de l'utilisateur pendante toute la phase d'enregistrement, jusqu'à ce que l'activation du *SigmaMenu* soit validée ou invalidée. Cette opération est gourmande en ressources processeur (complexité en $O(n^3)$) et a donc un impact important sur les performances de l'application pendant l'enregistrement. Pour réduire les temps de calcul, nous avons mis en place deux optimisations :

1. les points de contact trop proches les uns des autres ne sont pas pris en compte (nous avons utilisé un seuil de 5 pixels),
2. l'algorithme a été modifié afin de réutiliser une partie des calculs d'une itération à une autre. Cette optimisation a permis de réduire la complexité algorithmique de chaque itération en $O(n^2)$.

Il est également possible de ré-échantillonner les points du tracé afin qu'ils soient répartis à distances égales. Nos tests ont toutefois montré que cela réduisait la fiabilité du reconnaisseur.

D'autres types de reconnaisseurs, peuvent également être mis en place. En particulier il existe plusieurs autres algorithmes de régression circulaire comme la méthode des moindres carrés totaux ou réduits ou encore la méthode de KÅSA [217]. Celui choisi a l'avantage d'être assez facile à implémenter et de pouvoir être optimisé pour une reconnaissance itérative. Appliqué sur le jeu de données recueilli lors de notre étude pilote, il a fait preuve de résultats satisfaisants.

4.2.2 Mise en place du délimiteur de sélection

À la manière du *Control Menu* [169] et du *FlowMenu* [90], si la commande sert à activer un mode de contrôle continu, le *SigmaMenu* peut fusionner la sélection de cette commande avec le contrôle continu en un seul mouvement sur la surface. Une fois la sélection validée, le système bascule dans un mode où les mouvements servent contrôler le paramétrage associé à la commande.

Avec l'implémentation actuelle, les deux modes sont délimités par un délimiteur spatial à la manière du *Control Menu* de POOK et al. [169]. Nous travaillons toutefois à remplacer ce délimiteur par un délimiteur gestuel reposant sur la courbure du mouvement ou sur un geste *multitouch*.

Première méthode : délimiteur de sélection spatial

Avec le *Control Menu*, une distance limite – ou frontière – sert à délimiter la sélection de la commande du mode de contrôle continu. Cette frontière doit être suffisamment importante pour donner suffisamment de données au système pour déterminer la direction du mouvement tout en tenant compte des mouvements parasites du point de contact. Ces derniers étant toutefois assez petits, il est possible

de positionner la frontière assez proche du premier point de contact. Dans le cas du *SigmaMenu*, la frontière doit également marquer la séparation avec les mouvements circulaires servant à définir le sens de rotation. En effet, tant que le délimiteur de sélection n'est pas déclenché, l'utilisateur peut continuer à tourner autour du menu voire changer de sens de rotation avant de faire son choix. Ces mouvements circulaires ne sont pas parfaits et le rayon effectif varie tout au long du mouvement. Ainsi, comparativement au *Control Menu* une distance plus importante doit être parcourue avant la délimitation de la sélection. Un seuil de distance trop proche du rayon d'activation est susceptible de provoquer une sélection non désirée. Nous avons fait deux choix principaux concernant le calcul de cette distance : utilisation d'une distance fixe et suppression de la composante orthogonale à la direction de la commande.

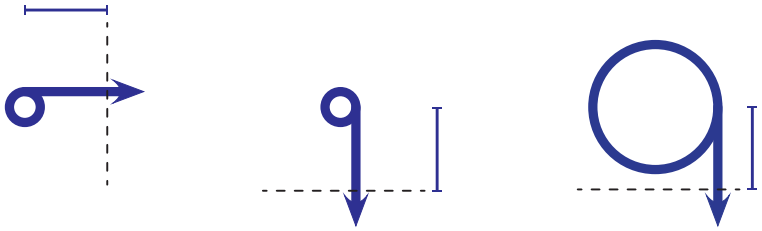
Distance fixe En théorie, plus le rayon du délimiteur circulaire est important, plus il est nécessaire d'utiliser une frontière éloignée. Toutefois, nous avons choisi de positionner cette frontière à une distance fixe, quel que soit le rayon du cercle servant de délimiteur d'activation. En effet, si la distance est variable, il est difficile pour un utilisateur d'anticiper le moment où le contrôle continu sera enclenché. Nous proposons d'utiliser une distance égale de 180 pixels correspondants à 1,71 cm sur un iPad Air.

Frontière perpendiculaire à la direction du mouvement Les premières versions du *SigmaMenu* se basaient sur la distance entre la position du premier contact et le centre du cercle d'activation pour déterminer la frontière avec le contrôle continu. La frontière était donc circulaire. Toutefois, plus le cercle d'activation était grand, plus le point de contact après l'activation finissait proche de la frontière et inversement. La longueur du mouvement pour délimiter la sélection dépendait donc du rayon du cercle d'activation. La Figure 4.13a représente graphiquement ce phénomène. La délimitation de la sélection était donc difficile à anticiper. De plus, lorsque l'activation circulaire était ample, il arrivait qu'un utilisateur dépasse la frontière par inadvertance et déclenche donc une sélection erronée.

Pour éviter ce problème, en fonction de la direction du mouvement (mise à jour pendant toute la durée du geste), la commande la plus proche est présélectionnée (la commande présélectionnée passe en noir sur le guide présenté en Figure 4.8). La composante orthogonale à la direction de cette commande est alors ignorée (Figure 4.13b). Par exemple, si la direction de la commande cible est parallèle à l'axe des abscisses, les mouvements parallèles à l'axe des ordonnées ne sont pas pris en compte. La frontière n'est donc plus circulaire, mais linéaire et perpendiculaire à la direction de la commande. De plus, en conséquence, sa position n'est pas fixe : la frontière se déplace lorsque la présélection est mise à jour (Figure 4.13b). La longueur du mouvement pour valider la présélection est toutefois toujours la même,

quel que soit le rayon du cercle d'activation. Selon nos tests, cette méthode s'est avérée beaucoup plus fiable.

(a) Frontière circulaire. La frontière est fixe, mais la longueur de la partie sélection du mouvement dépend du rayon.



(b) Frontière ignorant la composante orthogonale à la présélection. La longueur de la partie sélection du mouvement est constante quel que soit le rayon, mais la frontière se déplace en fonction de la direction présélectionnée.

Figure 4.13 – Différents types de frontière entre la sélection et le contrôle continu pour les *SigmaMenus*

Seconde méthode : délimiteur de sélection basé sur la courbure

De la même manière que l'invalidation du délimiteur d'activation peut être déclenchée lorsque la courbure est trop faible (voir la partie 4.2.1), au lieu d'utiliser un délimiteur de sélection spatial comme le *ControlMenus* et le *FlowMenu*, il est possible d'utiliser la détection d'un mouvement rectiligne.

Cette méthode est encore en cours d'expérimentation, mais peut permettre de réduire le nombre de sélections accidentelles lorsque la sélection est fusionnée avec le contrôle continu.

Troisième méthode : délimiteur de sélection *multitouch*

Alternativement : la séparation de la sélection et du contrôle continu peut être marquée par un tap sur la surface à l'aide d'un deuxième doigt.

L'avantage de cette méthode est qu'elle est susceptible de supprimer toute sélection accidentelle.

Son défaut est d'être davantage explicite et le geste est donc moins fluide. Elle aussi probablement moins rapide. Ainsi il n'est pas certain que ses performances soient meilleures que si le menu ne fusionnait pas le contrôle continu et où l'utilisateur à juste à relever le doigt et le reposer pour démarrer le contrôle.

4.2.3 Conception du guide

Les *SigmaMenus* introduisent plusieurs comportements originaux et un utilisateur novice a besoin d'être guidé pour apprendre à se servir de la technique.

Activation

Le premier guide intégré aux *SigmaMenus* permet d'indiquer le geste à effectuer pour activer la technique. Tracer un cercle pour activer une commande est inhabituel. Sur mobile, les utilisateurs sont davantage habitués à effectuer un appui long pour ouvrir un menu contextuel. Un délimiteur basé sur un délai est peu propice à une interaction rapide [141, 216] mais parait adapté pour permettre de découvrir les actions possibles et la façon d'interagir avec le système. Suite à la détection d'un appui long, nous faisons apparaître une petite animation sous la forme d'une flèche circulaire montrant le mouvement à effectuer pour activer le menu. Le sens de rotation indiqué par la flèche est inversé d'une fois à l'autre.

Sens de rotation

La plus grande difficulté lors de la conception du guide du *SigmaMenu* fut de montrer la dépendance au sens de rotation et donc la manière de sélectionner l'ensemble des commandes. Lors de l'ouverture du *SigmaMenu*, seulement la moitié des commandes est accessible. Cette moitié dépend du sens dans lequel a été tracé le délimiteur d'activation circulaire. Il est nécessaire d'indiquer que l'autre moitié des commandes est atteinte en changeant de sens de rotation autour du centre du menu.

La figure 4.14 montre la toute première itération du guide du *SigmaMenu*. La roue inactive (la roue antihoraire sur la figure 4.14) est plus fine et légèrement transparente. La forme en sigma de chaque flèche est censée montrer le mouvement à suivre pour sélectionner chaque élément. Toutefois, nos tests ont montré que ce guidage suggérait mal la façon de passer d'une roue à une autre et de sélectionner un élément. En effet, les utilisateurs avaient typiquement tendance à simplement se déplacer vers les étiquettes sans respecter les mouvements indiqués. Ceci avait pour conséquence une interprétation hasardeuse du sens de rotation et souvent, la sélection de la mauvaise commande.

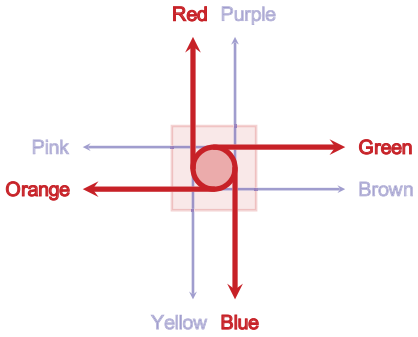


Figure 4.14 – Premier *design* du guide du *SigmaMenu*. Ce retour utilisateur retransmet mal l'influence du sens de rotation et le mouvement à effectuer pour accéder à chaque commande.

Différentes méthodes furent testées avant d'arriver au guide exposé à la Figure 4.8. Ce dernier implémente une métaphore de pivot bloquant. Lorsque l'on tourne dans le sens horaire (respectivement antihoraire), le guide suit le mouvement et pivote jusqu'à venir en butée au moment où la roue horaire (respectivement antihoraire) est activée. En parallèle et en suivant le même mouvement, l'autre roue est légèrement désaxée et désactivée. Un léger effet d'échelle, de flou et de transparence a également pour objectif de donner l'impression que la roue qui se désactive se dévisse et passe à l'arrière alors que l'autre se visse et passe à l'avant. Voir le menu pivoter en même temps que le mouvement met en évidence le lien entre les éléments accessibles et le sens de rotation. Le guide met également en avant la commande la plus proche en fonction de la direction du mouvement à un instant donné (voir la figure 4.8). Nos tests informels ont montré que les utilisateurs étaient tout d'abord intrigués par ce comportement inusuel, mais que celui-ci permettait effectivement une meilleure compréhension du fonctionnement du menu. Dans la littérature, on retrouve l'utilisation d'une métaphore 3D afin de réduire la complexité d'une technique pour les utilisateurs novices avec le *Wavelet menu* de Francone et al. [75].

4.2.4 Conceptions alternatives et extensions

Plusieurs approches ont été considérées lors de la conception du *SigmaMenu*.

Clock Menus

Le *Clock Menu* (voir la Figure 4.15) est une variante du *SigmaMenu* pour laquelle, la commande est sélectionnée en fonction de la position du premier point de contact sur le délimiteur d'activation circulaire. Par exemple, si l'utilisateur commence son cercle par la droite, une commande différente sera sélectionnée que s'il le

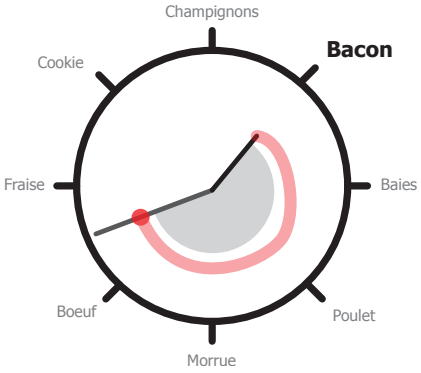


Figure 4.15 – Visuel expérimental pour le *Clock Menu*. La trace rouge représente les mouvements de l'utilisateur et le point rouge la position courante du contact. La petite aiguille indique l'angle de départ et la commande présélectionnée. La grande aiguille suit les mouvements du doigt et indique l'angle courant. Lorsque la grande aiguille repasse devant la petite aiguille, la sélection est validée.

commence par la gauche. La position de départ sur le cercle (et donc la commande présélectionnée) est indiquée par la petite aiguille sur la Figure 4.15.

Le délimiteur de sélection marquant l'activation de la commande et, le cas échéant, le début du contrôle continu n'est plus une distance à franchir, mais est marqué par la complétion du cercle. Ainsi, dans le cas de la sélection d'un mode permettant la manipulation continue d'un paramètre, dès qu'une révolution complète est tracée, le contrôle continu est enclenché. Cette méthode permet à l'utilisateur de contrôler plus précisément la position où le contrôle continu démarre puisque c'est lui même qui le positionne. Sur la Figure 4.15, la grande aiguille suit le déplacement des mouvements de l'utilisateur. Lorsqu'elle repasse sur la petite aiguille, le contrôle continu est donc enclenché.

Le défaut de cette méthode est que la commande sélectionnée est définie dès le début du mouvement et avant même l'ouverture du menu. Pour permettre à l'utilisateur novice de modifier son choix, nous proposons d'utiliser un délai : lorsque l'utilisateur marque une pause, la petite aiguille – qui marque la commande sélectionnée – est déplacée sur la grande aiguille, c'est-à-dire à l'angle correspondant à la position du doigt.

Un pilote où nous avons demandé a trois participants de tracer sans retour un cercle en commençant à un angle de départ donné a toutefois révélé une erreur moyenne non négligeable : 21,1 degrés, $\sigma = 13\,0$ degrés. Ce taux d'erreur est susceptible d'avoir un impact sur les performances d'un *ClockMenu* composé de 8 commandes différentes (soit une commande tous les 45 degrés). Par conséquent, cette technique n'a pas été retenue.

Multi-niveaux et métaphore de la vis

On peut imaginer accéder à d'autres menus en continuant de tourner une fois la technique activée. L'idée est de donner l'impression que l'on passe d'un niveau à un autre en « vissant » ou en « dévissant » le *SigmaMenu*. Un des intérêts principaux des mouvements circulaires est de ne pas être limités par les bords de l'écran [144]. Contrairement au *Marking Menus* par exemple, grâce à cette méthode, il est théoriquement possible d'accéder à un nombre de niveaux potentiellement illimité. Bien que nos prétests tendent à montrer que les utilisateurs peuvent effectuer un nombre de rotations donné autour d'un cercle (erreur moyenne sur 3 participants, sans entraînement et sans retour utilisateur de 0,29 révolution, $\sigma = 0{,}20$), nos premiers tests informels avec le *SigmaMenu* ont toutefois montré que cette méthode pouvait être difficile à utiliser en pratique. De plus, un guide efficace reste à concevoir.

4.2.5 Futurs travaux

Une étude est nécessaire pour déterminer quel couple délimiteur de sélection et délimiteur d'activation est le plus efficace.

En particulier, cinq points sont à considérer :

1. Taux d'erreur d'activation,
2. Taux d'erreur de sélection,
3. Précision de l'activation du mode par défaut (dans le cas où il est retardé),
4. Quantité de manipulation incidente (dans le cas où le mode par défaut est activé pendant l'enregistrement),
5. Précision de l'activation du contrôle continu (dans le cas où ce dernier est fusionné avec la sélection).

4.3 Étude écologique des techniques proposées au sein du prototype de radiologie

L'objectif de cette étude était de comparer les performances du *FingerMenu* et du *SigmaMenu* face à des techniques d'interaction classiques sur tablette : une barre d'outils et une barre de menus.

Le *FingerMenu* et le *SigmaMenu* sont des instances de techniques utilisant respectivement un délimiteur d'activation gestuel *multitouch* et un délimiteur d'activation gestuel mono-contact. Les barres d'outils et les barres de menu sont toutes deux des instances de techniques basées sur un délimiteur d'activation spatiale.

La barre d'outils est une technique d'interaction extrêmement rapide puisque les commandes sont affichées en permanence et un utilisateur a juste à toucher du doigt

le bouton associé pour en sélectionner une. Cependant, elle consomme beaucoup d'espace d'écran, en particulier sur tablette tactile où la taille des boutons doit être plus importante pour pallier le problème du « *fat finger* » [171, 220] abordé dans la Partie 2.1.4. Or, la surface d'affichage est précieuse sur un dispositif mobile comme une tablette tactile. Une barre de menus est assez semblable à une barre d'outils à l'exception que ses boutons ne permettent pas d'actionner une commande, mais d'ouvrir des menus linéaires qui eux contiennent les boutons permettant d'accéder aux commandes (ou a des sous-menus). Une barre de menus peut donc contenir beaucoup plus de commandes pour la même consommation de surface d'affichage, mais nécessite une étape supplémentaire par rapport aux barres d'outils. A contrario, le *FingerMenu* et le *SigmaMenu* ne consomment pas d'espace-écran tant que le menu n'est pas ouvert.

Le *FingerMenu* permet de sélectionner jusqu'à 5 commandes et est donc assez proche d'une barre d'outils en terme de taille de vocabulaire maximum. Nous pouvons cependant noter qu'il est possible de les étendre pour combiner plusieurs niveaux et ainsi atteindre facilement jusqu'à 50 commandes (Partie 4.1.4). Ces variantes n'ont toutefois pas été utilisées lors de cette expérience. Le *SigmaMenu* peut aller jusqu'à un vocabulaire de 16 commandes et s'approche donc des tailles de vocabulaire possibles avec une barre de menus.

Nous avons formulé cinq hypothèses :

H1 : Le *FingerMenu* permet de sélectionner une commande aussi rapidement qu'une barre d'outils (sans toutefois consommer d'espace-écran).

H2 : le *SigmaMenu* permet de sélectionner une commande aussi rapidement qu'une barre de menu (à nouveau sans consommer d'espace-écran).

H3 : Fusionner la sélection d'un paramètre avec son réglage permet de réduire le temps d'exécution totale (comportant activation, sélection et manipulation continue).

H4 : Les utilisateurs sont capables d'interagir sur la tablette en utilisant chacun de leurs doigts avec le *FingerMenu*.

H5 : Les utilisateurs peuvent aisément alterner entre les deux sens de rotation du *SigmaMenu*. Les résultats de nos prétests sur le délimiteur d'activation du *SigmaMenu* tendent déjà en faveur cette hypothèse (voir Partie 4.2.1).

Nous avons cherché à nous rapprocher le plus possible d'une tâche d'analyse radiologique réelle nécessitant d'alterner fréquemment entre différents modes de contrôle. Nous nous sommes placés dans le cadre du scénario de recherche d'anévrisme présenté en introduction du mémoire (Partie 1.2). Dans ce but, l'expérience a été implémentée directement dans le prototype de radiologie numérique développé dans le contexte de la thèse (voir la Partie 4.3.1). N'ayant pu avoir accès à des radiologues, nous avons toutefois dû adapter la tâche pour qu'elle puisse être exécutée par des participants non experts.

4.3.1 Développement du prototype

Le prototype (Figure 4.16) développé pendant la thèse a servi de support à cette expérience.

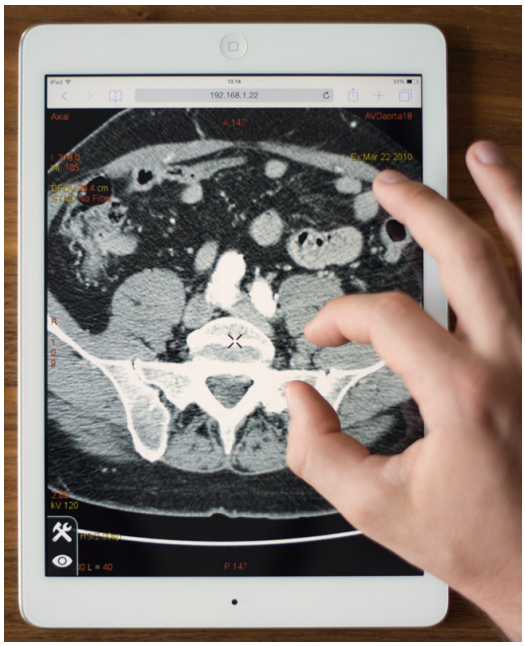


Figure 4.16 – Prototype de radiologie sur tablette. Le système permet actuellement la visualisation d'images radiologiques 3D sous forme de coupes, le déplacement dans les 3 dimensions, la rotation dans les 3 dimensions, le réglage de la luminosité et du contraste, l'agrandissement, et la transition vers différents types de vue (ici, vue axiale).

Ce prototype délègue l'ensemble des traitements et rendus visuels à une machine distante. Pour cela, il s'appuie sur un serveur capable d'exécuter et de communiquer avec des instances de *Volume Viewer*, le logiciel de radiologie développé par *GE HealthCare*. Seule la partie interactive est à la charge de la tablette qui reçoit du serveur un flux d'images (voir la Figure 4.17). Plusieurs raisons justifient ce choix d'architecture :

1. La puissance de calcul des tablettes actuelles est encore en deçà des performances requises pour le travail sur des images radiologiques.
2. *Volume Viewer* est un logiciel extrêmement complexe dont le début du développement remonte aux années 90. Porter le code existant sur la tablette n'est à priori pas possible et ré implémenter le moteur, même en partie, est inenvisageable considérant la complexité et la taille de ce dernier.

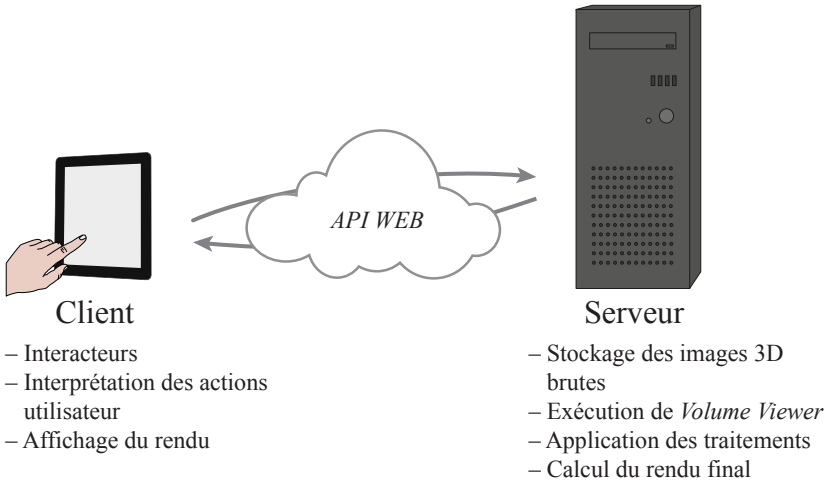


Figure 4.17 – Architecture du prototype

3. Les images radiologiques brutes peuvent être particulièrement volumineuses (parfois plusieurs gigaoctets). Les télécharger entièrement impliquerait une consommation réseau importante et un temps de transfert particulièrement long avant de pouvoir commencer à travailler sur un examen.

Afin de pouvoir intégrer facilement les briques logicielles développées pendant la thèse à un projet Internet de l'entreprise, il a été choisi d'implémenter le client entièrement en utilisant des technologies WEB (HTML & JavaScript). Celui-ci s'exécute donc directement à l'intérieur d'un navigateur Internet.

Le développement du prototype a été repris à la suite d'un projet interne terminé avant le début de la thèse. Toutefois, les technologies alors choisies (*Google Web Toolkit*[4] et *SmartGWT*[5]) ne permettaient pas d'exploiter les capacités *multitouch* de la tablette. Celui-ci a donc dû être presque entièrement réécrit pendant la thèse en HTML et JavaScript.

Le serveur WEB fait le pont entre *VolumeViewer* et le client. Il est capable de gérer les sessions et d'exécuter plusieurs instances de *VolumeViewer* ce qui permet à plusieurs utilisateurs de s'y connecter en parallèle.

Il communique avec *VolumeViewer* à l'aide de l'interface en ligne de commande du logiciel de radiologie (appelée *Voxtool Command Line*). La possibilité de répondre sous forme textuelle afin de communiquer de l'information depuis *VolumeViewer* vers le serveur a été ajoutée pour les besoins du projet (à l'origine la communication ne pouvait avoir lieu que dans un sens). De plus, plusieurs nouvelles commandes

4. *Google Web Toolkit* (GWT) est un ensemble d'outils logiciels *open sources* développés par Google et permettant la conception de sites WEB dynamiques entièrement en Java. La partie cliente est automatiquement traduite du Java au JavaScript.
5. *SmartGWT* est une surcouche de *GWT* proposant, notamment, un ensemble d'intéracteurs.

ont été implémentées (l'interface en ligne de commande est à l'origine prévue pour permettre l'exécution automatique de tests en interne et non pas pour contrôler entièrement le logiciel). Un script intégré au moteur de *VolumeViewer* se charge d'exporter le rendu sous forme d'images à chaque mise à jour de l'affichage afin qu'elles puissent être retransmises vers le client.

Le client (une application WEB exécutée sur la tablette) se charge de récupérer et afficher les images, d'interpréter les actions de l'utilisateur puis de les retransmettre vers le serveur sous forme de commandes. Il a été développé de manière à pouvoir facilement interchanger les techniques de menu, et ce même en cours d'exécution (un prérequis pour notre expérience).

4.3.2 Tâche et procédure

Afin de pouvoir réaliser des expériences avec des utilisateurs non spécialistes du domaine de la radiologie, nous avons tenté de « supprimer » la partie cognitive spécifique liée à une tâche de revue radiologique tout en conservant les spécificités interactionnelles qui la caractérisent.

Cinq pilotes successifs ont été exécutés. Chacun de ces pilotes nous a amené à revoir le protocole et le prototype expérimental afin d'augmenter la significativité des résultats. Le protocole décrit ci-dessous est celui qui a été utilisé pour le dernier de ces pilotes.

Une image 3D contenant cinq sphères concentriques – les unes à l'intérieur des autres à la manière de poupées gigognes – était présentée aux participants. Elles étaient de plus encerclées par un anneau de faible épaisseur (voir les figures 4.18, 4.19, 4.20, 4.21 et 4.22). Les sphères étaient de luminosité décroissante si bien qu'en fonction du réglage de luminosité et de contraste, l'ensemble ou une partie de ces sphères était visible. En complément, ce volume d'intérêt – sphères et anneau – était intégré à l'intérieur d'une image radiologique traditionnelle afin de simuler le bruit, mais aussi d'aider à se repérer dans l'image. Comme il est usuel lors d'une tâche de revue radiologique, l'image 3D était manipulée par le biais d'une coupe ce qui permet d'observer l'intérieur des volumes. Dans le cas contraire, seule la sphère extérieure serait visible puisque les autres sont « contenues » à l'intérieur.

Modes de manipulation. Les participants devaient manipuler 5 modes différents :

— *paging* (1D) qui permet de déplacer la coupe le long de son axe normal (Figure 4.19),

— *panning* (2D, aussi appelé *roaming* en radiologie) qui permet de déplacer la coupe dans son plan (Figure 4.20a),

— zoom (1D, Figure 4.20b)

— rotation centrée (2D) qui permet de faire tourner la coupe autour de son centre (la rotation autour de l'axe normal n'était pas possible, Figure 4.21),

— *windowing* (2D) qui permet de modifier la fenêtre de contraste c'est-à-dire d'ajuster les valeurs de contraste et de luminosité (Figure 4.22).

Objectifs. Le but de la tâche était de manipuler la coupe de la vue afin d'atteindre la position, la rotation et le niveau d'agrandissement où :

1. La coupe est dans le plan de l'anneau (celui-ci est donc entièrement visible à l'écran),
2. Le volume cible est centré,
3. Le volume cible occupe l'espace nécessaire (et est donc à la bonne échelle),
4. Les valeurs de contraste et de luminosité sont réglées de telle manière à ce que les cinq sphères concentriques soient visibles.

Les figures 4.18, 4.19, 4.20, 4.21 et 4.22 montrent les différentes étapes permettant de remplir ces objectifs.

Les quatre premiers modes devaient être sélectionnés grâce aux différentes techniques (*FingerMenu*, *SigmaMenu*, barre de menus ou barre d'outils) alors que le zoom devait être effectué en utilisant un geste de *pinch* avec deux doigts. Chaque technique permettait d'atteindre jusqu'à 16 éléments et de fausses commandes étaient disposées aux positions non utilisées. Les commandes étaient positionnées aux positions les plus efficaces : dans le menu le plus à gauche pour la barre de menu et à droite avec le *SigmaMenu*. Le *FingerMenu* et la barre d'outils, quant à eux, n'étaient équipés que d'un seul niveau.

Guides. Pendant la rotation, le *pan* et le zoom, un cercle jaune au centre de la vue indiquait la destination et le niveau de zoom requis. Lorsque le réglage cible était atteint, ce cercle passait au vert. Pendant le *paging* et le *windowing*, une flèche jaune indiquait la direction que devait avoir le mouvement pour obtenir le réglage cible. Il suffisait donc de la suivre pour atteindre ce réglage. Lorsque celui-ci était atteint, la flèche se transformait en disque vert. Ce guide permet de supprimer la partie cognitive de la tâche originale tout en conservant sa nature. Il évitait par exemple que les participants, peu familiers avec la manipulation d'une vue-coupe d'un volume 3D, ne « se perdent ».

Positions de départ. Le volume d'intérêt pouvait prendre huit positions (et degré de rotation) initiales différentes. Celles-ci furent choisies à égale distance de la cible et de manière à ce que la manipulation des 5 modes soit nécessaire pour compléter un essai.

4.3.3 Résultats et conclusions

Cinq pilotes successifs ont été exécutés, chacun rassemblant entre 2 et 3 participants et basé sur un protocole et un prototype expérimental différent. Celui

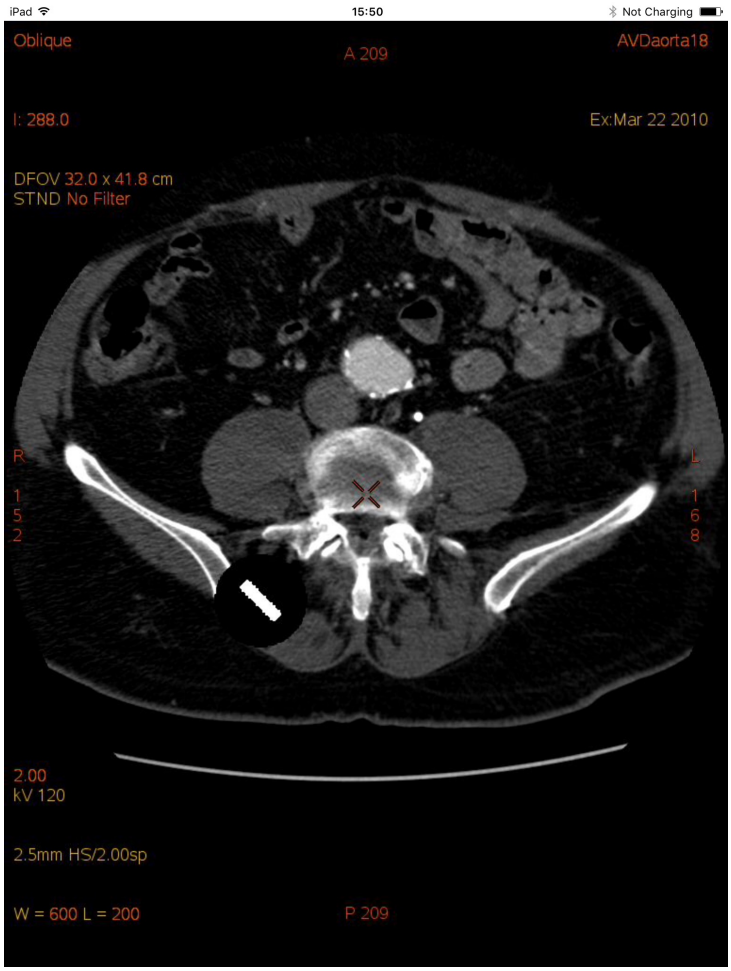
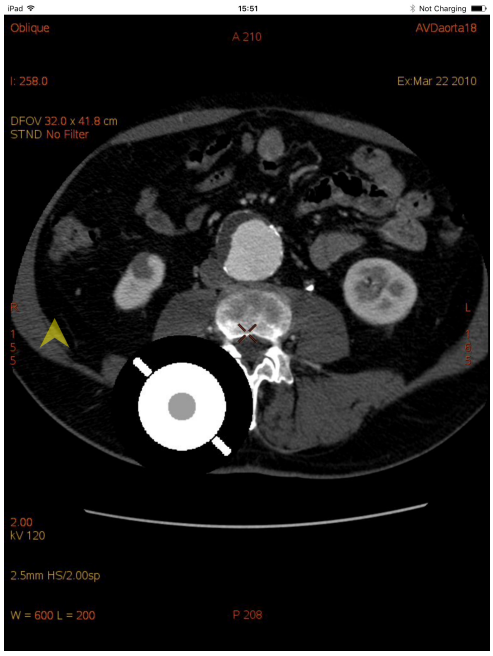
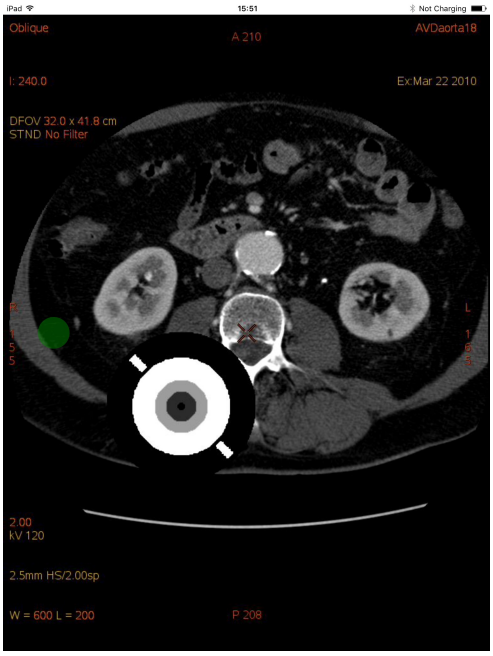


Figure 4.18 – Vues initiale lors de l'étude au sein du prototype de radiologie. La cible apparaît seulement comme un rectangle incliné à 45 degrés au centre d'un disque noir excentré en bas à gauche. Il s'agit en fait de l'extrémité de l'anneau encerclant les sphères.

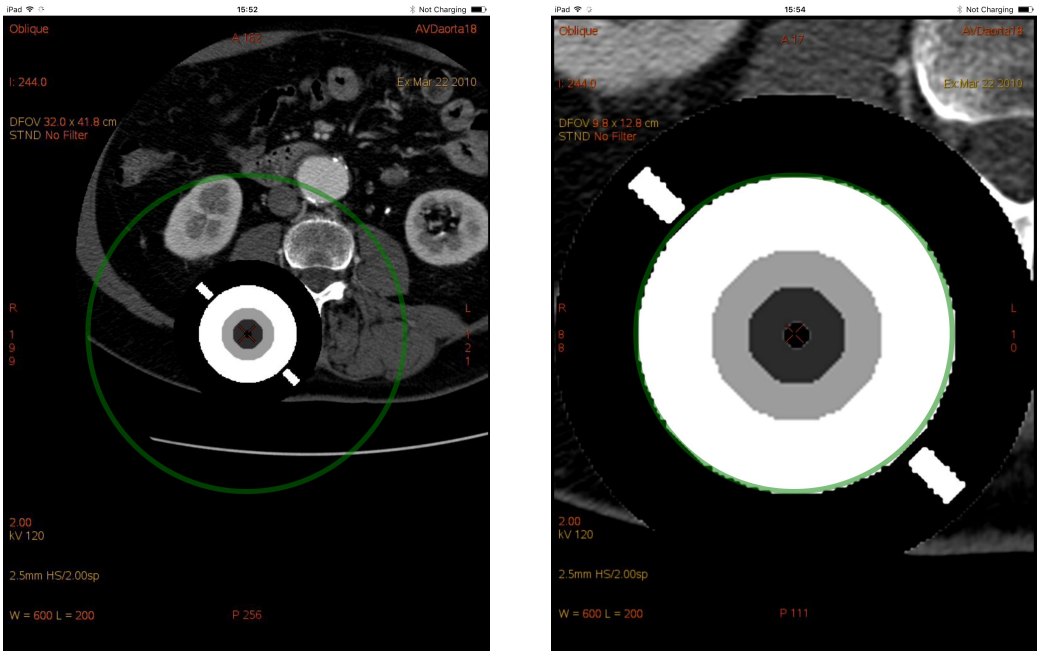


(a) *Paging* (translation sur l'axe perpendiculaire à la vue). La flèche jaune à gauche indique la direction du mouvement.



(b) Fin du *paging*. La flèche jaune s'est transformée en disque vert pour indiquer que la position cible a été atteinte. Le plan de la vue passe au centre des sphères. Il est de plus perpendiculaire au plan de l'anneau, celui-ci n'apparaît donc que comme deux rectangles sur le côté.

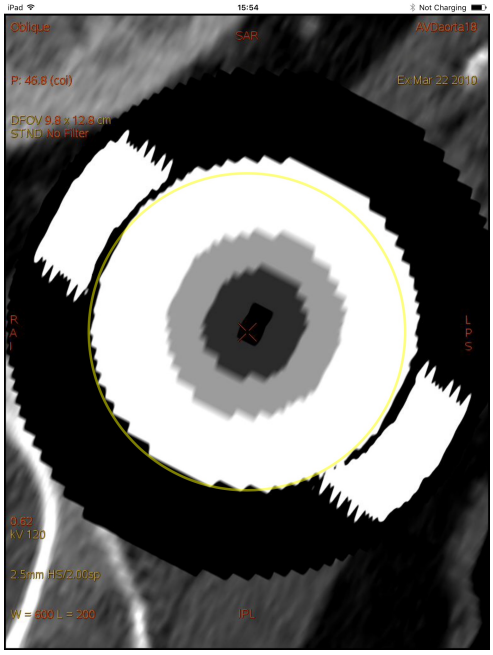
Figure 4.19 – *Paging* lors de l'étude au sein du prototype de radiologie.



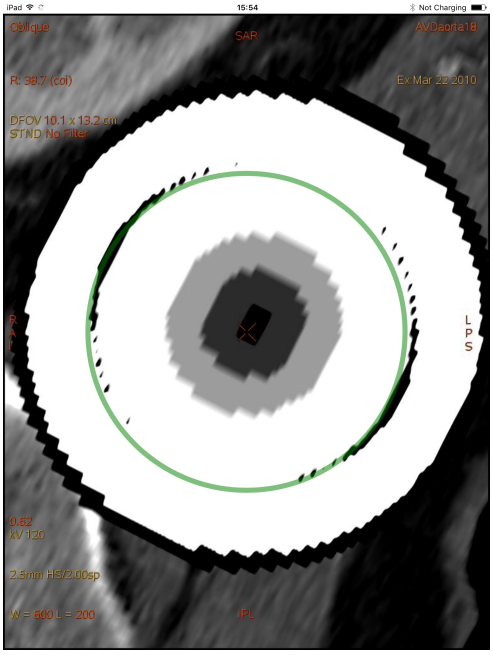
(a) Fin du *panning* (déplacement dans le plan de la vue). Le volume cible a été déplacé au centre de la vue. Le guide circulaire est passé au vert car la position cible est atteinte.

(b) Fin du *zoom*. Les bords de la sphère sont au niveau du guide circulaire.

Figure 4.20 – *Panning* et zoom lors de l'étude au sein du prototype de radiologie.



(a) Début de la rotation. Jusqu'ici, la vue était perpendiculaire au plan de l'anneau. Celui-ci n'apparaissait donc que comme deux rectangles sur les côtés (voir Figure 4.20b par exemple). Ici, il commence à apparaître plus entièrement.



(b) Fin de la rotation. La coupe est à présent dans le plan de l'anneau et celui-ci est donc entièrement visible.

Figure 4.21 – Rotation lors de l'étude au sein du prototype de radiologie.

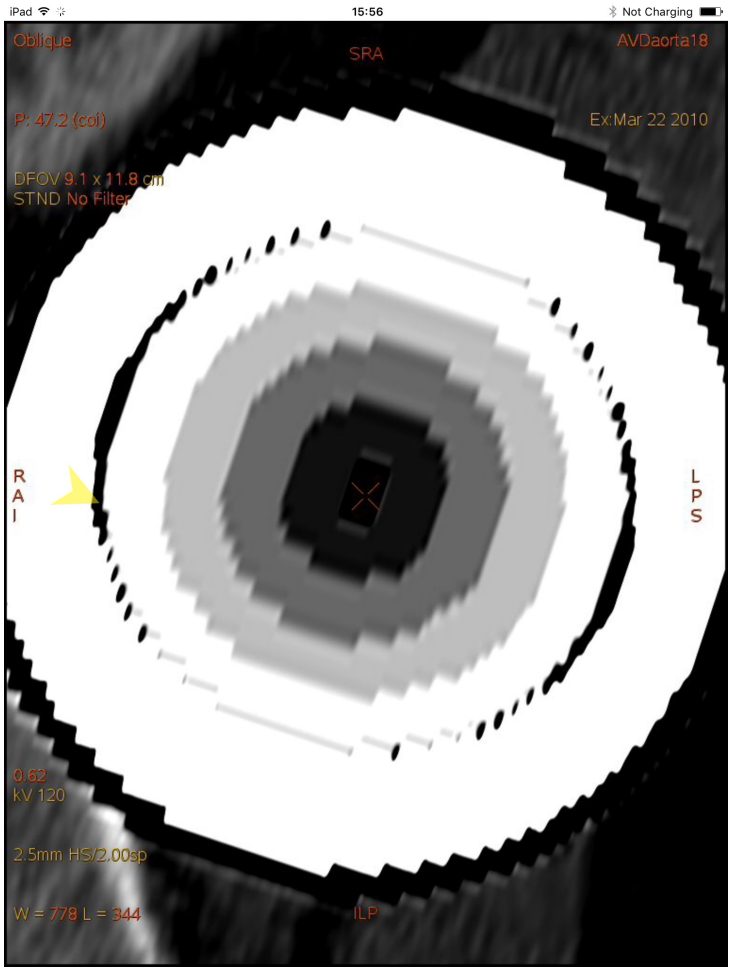


Figure 4.22 – *Windowing* et vues finales lors de l'étude au sein du prototype de radiologie. Le *windowing* a permis de faire apparaître les cinq sphères concentriques en ajustant le contraste et la luminosité de l'image.

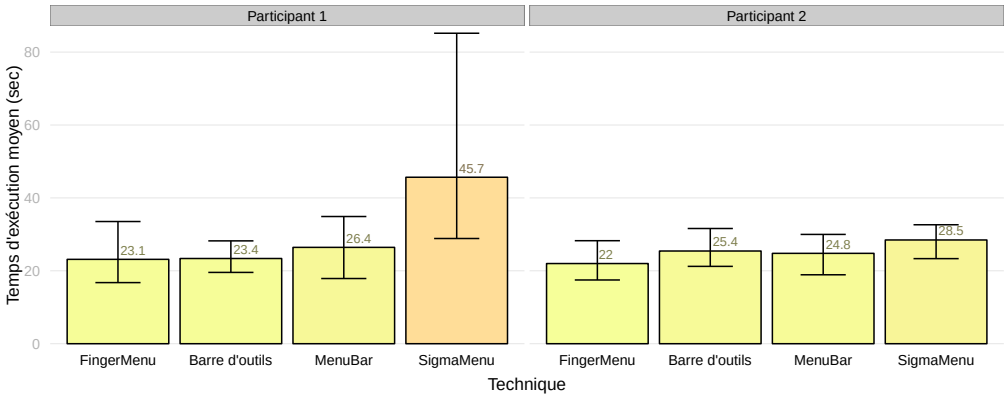


Figure 4.23 – Résultat du dernier pilote pour l'étude écologique

présenté ci-dessus est celui qui a été utilisé lors du dernier pilote. Les résultats de ce pilote sont donnés en Figure 4.23.

Il s'avère qu'une trop grande partie du temps fut consacré au réglage des différents paramètres (position, rotation, niveau de zoom et *windowing*) par rapport au temps de sélection. De plus, cette étape était touchée par une importante variabilité malgré les guides progressivement ajoutés à l'expérience ce qui limite grandement la significativité des résultats obtenus.

À la suite des ce dernier pilote, nous avons finalement décidé d'abandonner l'idée de faire une expérience au sein du logiciel de radiologie de *General Electric*. La tâche, bien que proche d'une utilisation réelle, s'est montrée particulièrement longue à implémenter et très difficile à mettre en place. Elle s'est avérée peu propice à une mesure précise des performances d'une technique de sélection via une expérience contrôlée en laboratoire du fait d'un temps de manipulation particulièrement long et variable ce qui rendait difficile d'obtenir des différences significatives.

4.4 Étude formelle

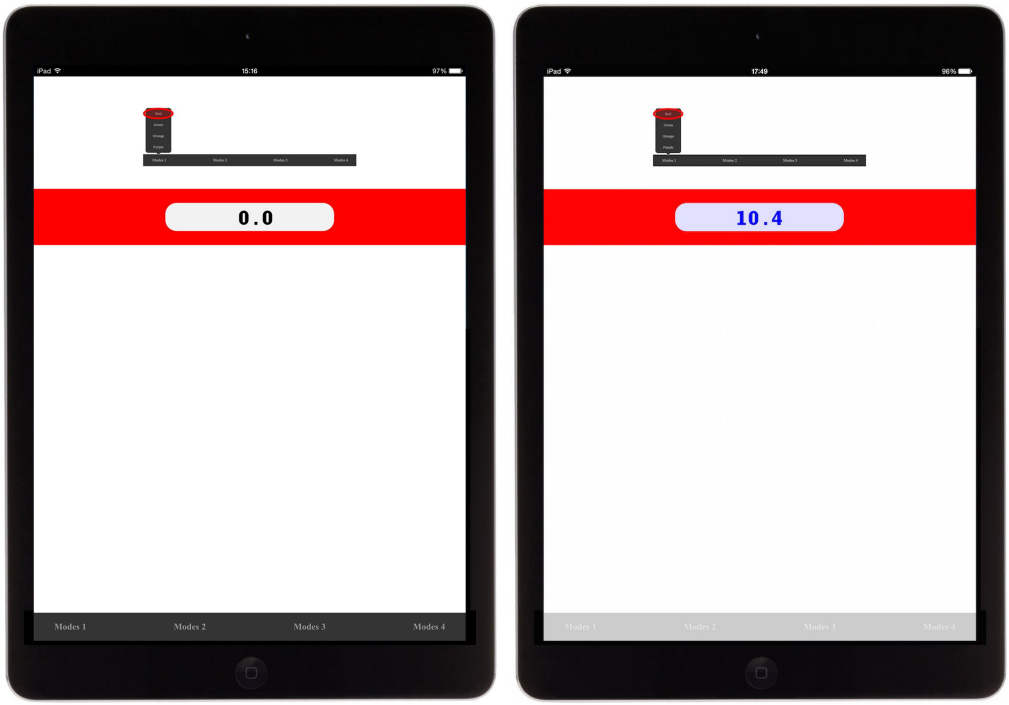
Cette seconde expérience reprend la précédente avec une tâche abstraite plus facile à contrôler. Les mêmes techniques furent comparées (voir la Partie 4.3 pour plus de détails) :

— Le *SigmaMenu* utilisant un délimiteur d'activation gestuel mono-contact,

— Le *FingerMenu* utilisant un délimiteur d'activation gestuel *multitouch*

— La barre de menus et la barre d'outils utilisant toutes deux un délimiteur d'activation spatiale.

Nous formulons également les mêmes hypothèses.



(a) Début d'un essai (b) Fin d'un essai

Figure 4.24 – Essais avec la barre de menu lors de l'étude formelle. Le stimulus est affiché en haut (les différents stimulus utilisés sont représentés en Figure 4.25). La couleur de la boîte autour du compteur représente le mode cible. Le compteur prend la couleur du mode sélectionné une fois que la bonne distance a été atteinte (ici à droite : bleu).

4.4.1 Participants et équipement

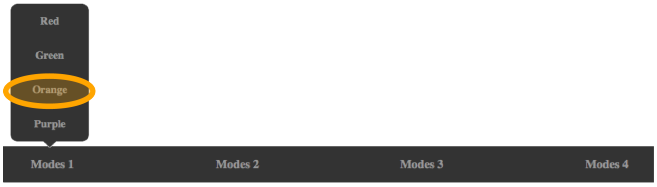
16 participants (dont 5 femmes) âgés de 23 à 32 ans (moyenne d'âge 26,8 ans, $\sigma = 2{,}4$ ans) ont été recrutés. Ils reçurent une poignée de bonbons pour leur participation. Les techniques et la tâche furent programmées en JavaScript et exécutées sur un Apple iPad Air.

4.4.2 Tâche et procédure

La tâche consistait à sélectionner un mode et à contrôler une valeur de 0 à 10 (voir la Figure 4.24). Il était demandé aux participants de la réaliser le plus rapidement possible tout en minimisant les erreurs.

(a) Exemple de stimulus pour le *FingerMenu*

(b) Exemple de stimulus pour le *SigmaMenu*



(c) Exemple de stimulus pour la barre d'outil

Figure 4.25 – Exemples de stimulus lors de l'expérience formelle

Stimulus et sélection. Chaque mode était représenté par une couleur différente. Le stimulus était affiché sur le haut de l'écran. Afin de simuler le comportement d'un expert, il indiquait exactement le geste à effectuer pour sélectionner le mode cible à l'instar, par exemple, de l'expérience de Balakrishnan et Patel sur la *PadMouse* [23]. La Figure 4.25 montre les stimulus utilisés pour les différentes techniques.

Une seule sélection par essai était possible. Il était demandé aux participants de finir la tâche (et donc continuer avec la phase de contrôle de la valeur) même si le mauvais mode avait été activé.

Phase de contrôle. Pendant la phase de contrôle, un compteur affichait une valeur correspondant à la distance parcourue (Figure 4.24b). Une fois le mode sélectionné, les participants devaient atteindre une valeur comprise entre 10 inclus et 11 exclu. Ceci correspondait à un déplacement d'environ 3,7 cm à 4,1 cm (il était possible de revenir en arrière). De manière à ne favoriser aucune technique, la direction du déplacement n'était pas prise en compte.

Début et fin d'un essai. Afin que la main du participant soit toujours à la même position au début de chaque tâche, celui-ci devait presser un bouton de démarrage au centre de l'écran avant chaque essai. Une fois le stimulus apparu, les participants étaient invités à effectuer les actions correspondantes (sélection de la commande

puis contrôle continu) aussi rapidement et précisément que possible. La tâche était achevée dès que l'utilisateur retirait le doigt lors de la phase de contrôle continu.

Retour. À chaque essai, la couleur de la boîte indiquait le mode cible. Par exemple, rouge sur la Figure 4.24a.

Par défaut, le compteur était noir. Toutefois, une fois le réglage cible atteint, il prenait la couleur sélectionnée ce qui permettait de savoir si le bon mode avait été activé. Sur l'exemple de la Figure 4.24b, le mode cible était « rouge » (couleur de la boîte) alors que le mode sélectionné fut « bleu » (couleur du compteur). Deux raisons ont motivées ne faire apparaître ce retour qu'une fois les valeurs cibles atteintes (et donc après un déplacement suffisamment long). D'une part cela donnait un retour supplémentaire lorsque la zone de succès était rejointe. D'autre part, on évitait alors que les participants ne soient tentés de terminer l'essai sans finir la tâche lorsque le mauvais mode avait été sélectionné (un comportement contraire aux instructions, mais néanmoins observé lors de nos pilotes).

Répartition des modes. Le prototype expérimental comportait cinq modes différents (c'est-à-dire cinq couleurs : rouge, verre, orange, violet et bleu)

Quatre de ces modes étaient groupés au même endroit alors que le dernier était à part. La raison principale de ce choix est qu'avec le *FingerMenu*, le pouce est un doigt à part et nous souhaitions reproduire ce phénomène avec les autres techniques. Dans le cas du *SigmaMenu*, les quatre modes « principaux » étaient groupés sur la même roue alors que le dernier était accessible avec l'autre sens de rotation. Dans le cas de la barre de menus, quatre modes étaient contenus dans le menu le plus à gauche alors que le dernier se situait dans le deuxième menu.

4.4.3 Protocole

Un protocole à mesures répétées intra participants fut utilisé. La variable indépendante était Technique (barre de menu, barre d'outils, *SigmaMenu*, *FingerMenu*).

L'expérience était composée de 4 parties distinctes, chacune dédiée à la manipulation d'une technique en particulier. L'ordre des techniques était balancé par un carré latin. Chaque partie était divisée en 4 blocs de 40 essais ce qui donne un total de 16 participants $\times$ 4 techniques $\times$ 2 blocs $\times$ 40 essais $=$ 5120 sélections mesurées. Les participants disposaient également de 32 sélections non mesurées avant chaque partie ce qui leur permettait de se familiariser avec les techniques. Ils étaient autorisés et encouragés à prendre une pause entre chaque bloc. L'expérience durait une heure au total.

4.4.4 Résultats et discussion

Temps d'exécution

Le temps d'exécution est mesuré entre le moment où le participant lève le doigt du bouton de démarrage jusqu'au moment où il lève le doigt à la fin de la manipulation du compteur.

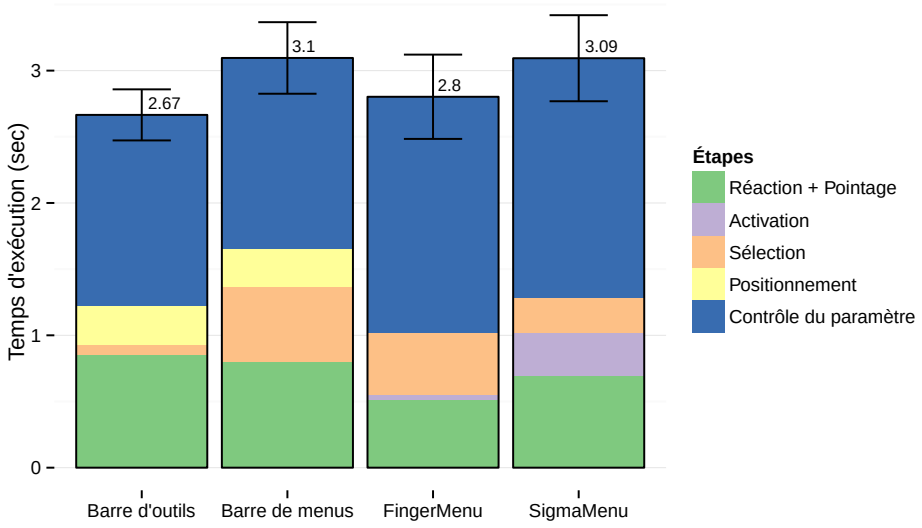


Figure 4.26 – Temps d'exécution

Une ANOVA a montré un effet significatif de la technique sur le temps d'exécution ($F_{3,45} = 15{,}95$, $p < 0{,}001$, voir Figure 4.26). Un *pairwise t-test* n'a révélé une différence significative qu'entre deux groupes de techniques : le *FingerMenu* et la barre d'outils d'une part, la barre de menus et le *SigmaMenu* d'autre part.

Cela tend donc à montrer que les *SigmaMenu* et les *FingerMenu* réussissent à atteindre des performances équivalentes aux techniques traditionnelles de même gabarit (la barre d'outils capable d'accéder à une petite dizaine de commandes pour le *FingerMenu* et la barre de menus pour le *SigmaMenu*) sans toutefois consommer d'espace-écran.

Temps de sélection total

Le temps de sélection total est calculé entre l'apparition du stimulus et la sélection effective d'une commande. Il intègre donc le temps de réaction, de pointage, d'activation et de sélection, mais pas le temps de positionnement et de contrôle (voir Figure 4.27).

Une ANOVA a montré un effet significatif du facteur Technique sur le temps de sélection total ($F_{3,45} = 45\,9$, $p < 0{,}001$). À nouveau, un *pairwise t-test* n'a révélé

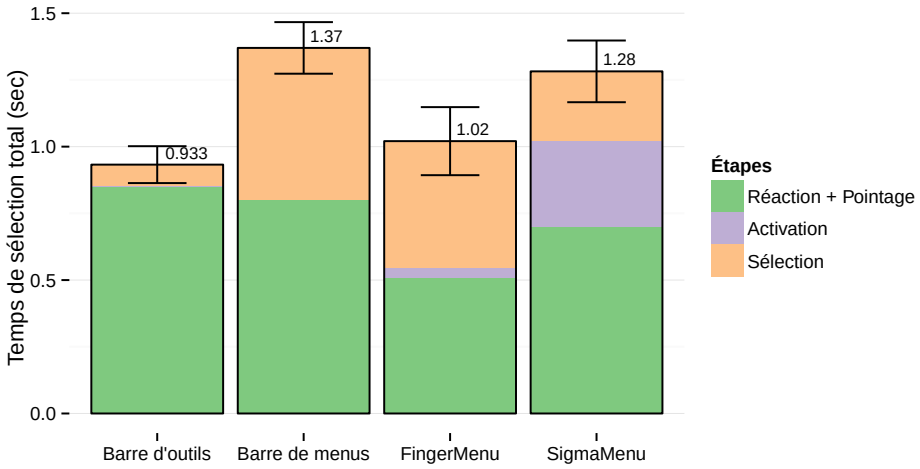


Figure 4.27 – Temps de sélection total

une différence qu'entre deux groupes de techniques : le *FingerMenu* et la barre d'outils d'une part, la barre de menus et le *SigmaMenu* d'autre part.

Cela tend donc à confirmer nos hypothèses H1 et H2 : le *FingerMenu* permet de sélectionner une commande aussi rapidement qu'une barre d'outils, et le *SigmaMenu* aussi rapidement qu'une barre de menu.

On peut toutefois noter que lors de cette expérience, les deux techniques fusionnaient la sélection de la commande avec le contrôle continu. Dans le cas contraire, le délimiteur de sélection est parfois différent (par exemple la sélection peut n'être confirmée qu'une fois le doigt relevé et non après avoir dépassé un seuil de déplacement). Une étude supplémentaire est nécessaire pour évaluer les performances en sélection sans fusion de la sélection avec le contrôle continu.

Temps de contrôle et temps de positionnement

Le temps de positionnement est mesuré entre l'instant où la commande est sélectionnée et le moment où le contrôle continu commence. Notez qu'il n'y a pas de temps de positionnement pour le *SigmaMenu* et le *FingerMenu* puisqu'ils fusionnent la sélection et la manipulation directe. Le contrôle continu commence donc immédiatement après la sélection de la commande avec ces deux techniques.

Une ANOVA a montré un effet significatif de la technique sur le temps de contrôle ($F_{3,45} = 22{,}04$, $p < 0{,}001$). Le temps de contrôle a été plus court pour la barre d'outils et la barre de menus. Cependant, étonnamment et comme le montre la Figure 4.28, le temps de positionnement compense parfaitement cette différence et une ANOVA n'a montré aucune différence significative pour la somme du temps de positionnement et du temps de contrôle.

Ce résultat considéré conjointement avec le temps d'exécution tend à invalider

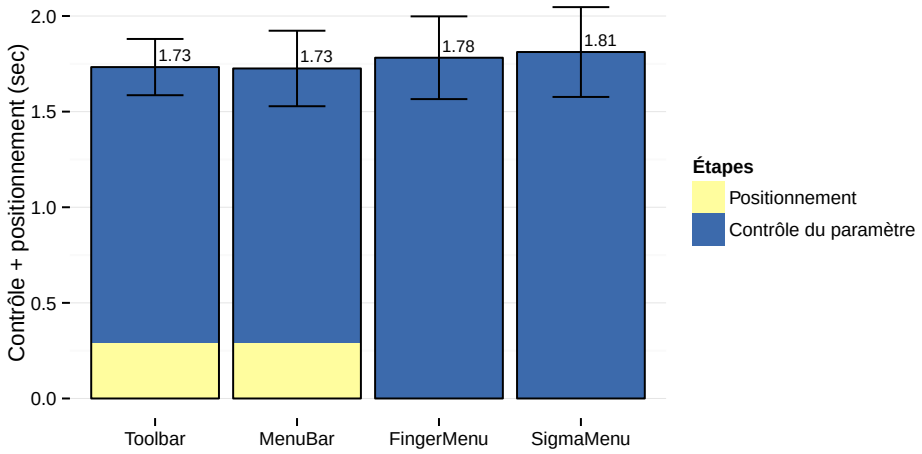


Figure 4.28 – Temps de contrôle et de positionnement

notre hypothèse H3 : fusionner la sélection et le contrôle du paramètre ne semble pas avantager les performances du *FingerMenu* et du *SigmaMenu* malgré la suppression de l'étape de repositionnement.

Ceci est surprenant si on considère le succès des précédentes techniques ayant utilisées cette approche (notamment le *Control Menu* [169] et le *Flow Menu* [90]). Une explication pourrait être que le contrôle de l'outil implique une charge cognitive qui peut être anticipée lors du repositionnement. Sur une tablette, le mouvement de repositionnement peut ne pas être suffisamment long pour rattraper le temps consommé par cette charge cognitive. Toutefois, il faut noter que lors de cette expérience, afin de ne pas défavoriser la barre d'outils et la barre de menus, les utilisateurs pouvaient commencer le contrôle du paramètre à partir de n'importe quelle position sur la tablette, et pouvaient aller dans n'importe quelle direction pour le régler. Il était donc possible de reposer le doigt à proximité du bouton ayant servi à sélectionner la commande, soit un mouvement de repositionnement minimal. Il est probable que le résultat soit différent avec une tâche demandant davantage de repositionnement, par exemple s'il est nécessaire de manipuler le réglage avec un déplacement de haut en bas (les barres d'outils ou de menus sont manipulées depuis le bas de l'écran).

Précision

Une ANOVA a montré que l'erreur de sélection a été significativement impactée par le facteur Technique ($F_{3,45} = 11{,}26$, $p < 0{,}002$ après correction de sphéricité, voir Figure 4.29). Un *pairwise t-test* n'a toutefois révélé une différence significative seulement entre le SigmaMenu et les autres techniques.

Aucune différence significative n'a été constatée sur le nombre d'essais finissant

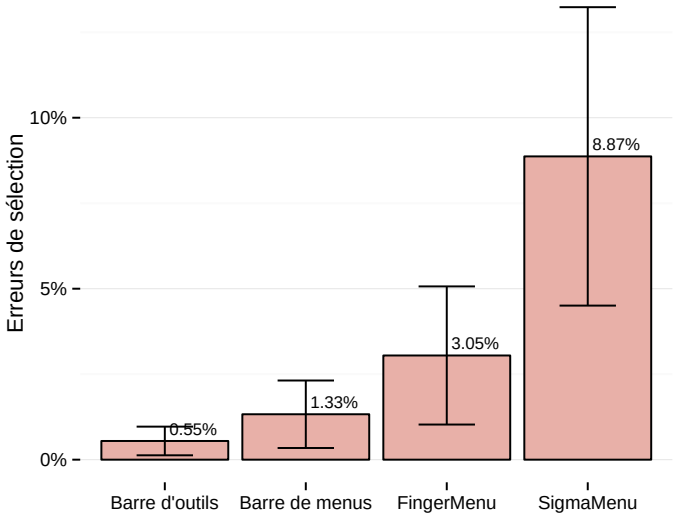


Figure 4.29 – Erreurs de sélection

en dehors de la zone de succès (déplacement trop long ou trop court). En moyenne, 94,3 % des essais ont été terminés dans la zone de succès.

Variations intra techniques

FingerMenu. La figure 4.30 représente le temps d'exécution pour chaque doigt avec le *FingerMenu*. Une ANOVA n'a relevé aucun effet significatif bien qu'une tendance semble se dessiner sur la figure 4.30. Ce résultat est donc en faveur de l'hypothèse H4 : les utilisateurs peuvent interagir sur la tablette avec chacun de leurs doigts avec le *FingerMenu*.

Une hypothèse pouvant expliquer cette faible différence peut être que lors des mouvements sur une tablette, la plupart des articulations engagées sont surtout celles du bras et du poignet. Les mouvements des doigts sont moins utilisés, car ils ne suffisent de toute façon pas à couvrir la taille de la surface d'interaction. L'interaction gestuelle avec un dispositif de plus petite taille, comme un téléphone, est en théorie consistuée de mouvements plus petits et donc impliquant une plus grande précision. Il est probable que dans ce cas, les articulations des doigts soient davantage mises à profit et que donc la différence entre doigts soit accentuée. Lors du chapitre 6 nous nous intéresserons plus en détail aux performances entre doigts dans le contexte du pointage sur téléphone. Les résultats de cette étude confirment la tendance observée sur la figure 4.30 en mettant en évidence une différence significative entre doigts.

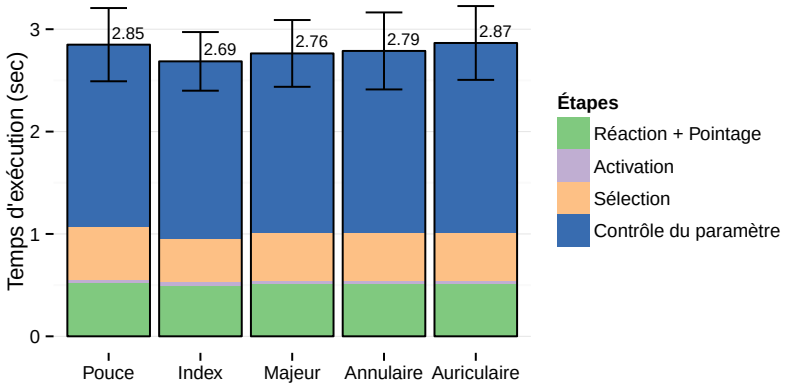


Figure 4.30 – Temps d'exécution pour chaque doigt avec le *FingerMenu*

SigmaMenu. Une ANOVA n'a relevé un effet significatif ni sur la direction de la commande, ni le sens de rotation. Ce résultat semble donc confirmer ceux de notre pilote indiquant que les utilisateurs n'avaient pas de difficulté particulière pour tracer un cercle dans un sens ou dans un autre. Il tend donc à valider l'hypothèse H5 : les participants peuvent exploiter ce degré de liberté sans difficulté.

Barre de menu. La position du bouton a eu un effet sur le temps d'exécution pour la barre de menu ($F_{3,45} = 13{,}32$, $p < 0{,}001$) et un *pairwise t-test* a montré que la sélection de l'élément positionné dans le second menu était significativement plus lente que pour les autres.

4.5 Conclusion et perspectives

Nos résultats suggèrent que les techniques utilisant des délimiteurs d'activation spatiaux restent bien entendu une approche efficace en terme de rapidité d'exécution. La barre d'outils, par exemple, reste une référence difficile à égaler. Les utilisateurs sont de plus très familiers avec cette technique et donc fortement entraînés à sa manipulation. Toutefois, ces hautes performances se font au prix d'une consommation importante d'espace d'affichage. La taille des boutons a par ailleurs une grande influence. Nous verrons lors du chapitre 6 que les performances se dégradent fortement si on la diminue. Sur tablette, cela pose particulièrement problème, car la ressource espace-écran est rare et son utilisation doit être optimisée pour être consacrée autant que possible à l'affichage des objets d'intérêt.

Le *FingerMenu* permet des performances similaires à une barre d'outils tout en n'occupant pas l'écran. Par défaut, il permet d'accéder jusqu'à 5 commandes différentes (une pour chaque doigt), mais il peut facilement être étendu pour 25 voire 50 en utilisant les variantes multi niveaux que nous avons proposées.

Le *SigmaMenu* est un peu plus lent, mais permet par défaut d'accéder jusqu'à 16 commandes. Sur tablette tactile, ceci est difficile avec une barre d'outils sans rajouter une seconde rangée de boutons ce qui double l'espace d'affichage occupé. Le *SigmaMenu* a fait preuve de performances équivalentes à une barre de menu mais, à l'instar du *FingerMenu*, ne consomme pas d'espace-écran.

Cette première étude ouvre de nombreuses perspectives.

SigmaMenu. Dans un premier temps nous continuons à améliorer le *SigmaMenu*. Nous cherchons notamment à utiliser la courbure plutôt qu'une distance pour invalider le délimiteur d'activation, mais aussi pour délimiter la sélection. Notre hypothèse est que ces nouveaux délimiteurs permettront de réduire la durée de l'enregistrement de la technique (en particulier lors de l'invalidation de son délimiteur d'activation), et d'améliorer son taux d'erreurs. Une expérience sera bien entendu nécessaire pour valider cette nouvelle approche.

Nous souhaitons également exécuter une étude comportant 16 commandes afin de mieux exploiter les capacités du menu. Dans un premier temps, nous envisageons de nous concentrer sur une tâche de sélection pure, sans contrôle continu.

De plus, bien que cette première étude semble indiquer que les deux sens de rotation peuvent être utilisés à performances égales, il n'est pas certain que l'exploitation de ce degré de liberté n'ait pas d'impact sur les performances. Nous envisageons d'explorer davantage cette hypothèse, notamment grâce à une expérience comparant le *SigmaMenu* avec une variante qui serait indifférente du sens de rotation.

Nous souhaitons également étudier le facteur mémorisation.

FingerMenu. Le *FingerMenu* s'est également avéré particulièrement prometteur. Différentes variantes ont été proposées afin d'accroitre la taille des vocabulaires de commandes possibles par exemple en y adjoignant des gestes directionnels inspirés du *Marking Menu*. D'autres expériences doivent être exécutées pour évaluer les capacités de ces extensions.

Fusion de la sélection et du contrôle continu Enfin, étonnamment, la capacité du *FingerMenu* et du *SigmaMenu* à fusionner la sélection de commandes et le contrôle continu à la manière du *Control Menu* [169] ne semble pas les avoir favorisés durant cette étude. Les raisons de ce résultat sont peut-être dues au protocole expérimental qui, pour ne pas défavoriser la barre d'outils et la barre de menus, ne contraignait pas le sens ou la position de départ du déplacement sur la surface servant à contrôler le paramètre à régler. Une étude supplémentaire est nécessaire pour valider cette hypothèse.

5 Support des utilisateurs experts et raccourcis

Il est important pour le concepteur en IHM d'anticiper la transition qui va progressivement conduire l'utilisateur, par un effet d'apprentissage, du statut de novice à celui d'expert. Dans ce chapitre nous allons nous intéresser au cas particulier des experts, des utilisateurs ayant acquis par la pratique une connaissance avancée des différentes fonctionnalités d'une interface. Ce facteur de familiarité doit naturellement être pris en compte dans le travail de conception : l'interface doit permettre la rapidité d'exécution exigée par ce type d'utilisateurs.

Dans le contexte de l'ordinateur standard, une stratégie éprouvée consiste à proposer des *raccourcis* clavier, combinaisons de touches ou touches spéciales, en complément des menus linéaires et des barres d'outils. Bien qu'en pratique la transition s'effectue tardivement et seulement pour un nombre restreint de commande, ces raccourcis accélèrent grandement l'accès aux fonctionnalités les plus fréquentes [52, 59, 85, 143].

Mais la tablette tactile est rarement couplée à un clavier physique, de sorte que la plupart du temps on ne saurait songer aux raccourcis clavier. Bien qu'encore rarement mis en place (voir la Partie 2.3), les gestes ont le potentiel d'être une technique de raccourcis efficace [7]. Les *Marking Menus*, notamment, se sont imposés comme la référence des menus gestuels sur tablette, notamment grâce à l'intégration d'un raccourci gestuel qui s'apprend à travers l'utilisation du menu : le même mouvement est utilisé dans les deux cas ce qui limite le coup d'un changement de modalité [59, 126, 128, 196].

Nous allons nous intéresser aux techniques de raccourcis à destination des utilisateurs experts. Tout d'abord, nous présenterons les Lettres Augmentées, une technique d'interaction de notre conception qui permet l'accès à un nombre important de commandes tout en étant plus facile à mémoriser que le *Marking Menu* grâce à l'utilisation du langage naturel et de l'écriture (Partie 5.1). Nous présenterons ensuite la Barre de Menus *multitouch*, une technique d'interaction qui prend une approche différente du *Marking Menu* et la plupart des techniques d'interaction gestuelle en conservant la *rétrocompatibilité* avec les techniques d'interaction traditionnelles, et en s'intégrant directement au sein d'une barre de menu (Partie 5.2).

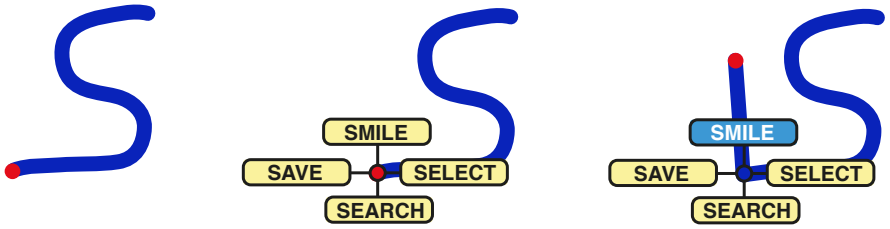


Figure 5.1 – Exemple de Lettres Augmentées en mode novice. Le menu apparaît après un délai de 500 ms une fois la lettre tracée.

5.1 Lettres Augmentées

Le *Marking Menu* [126, 128] (abordé en 2.3.4) est un menu circulaire où un utilisateur exécute une commande en traçant un trait dans la direction du nom de cette commande. La sélection est particulièrement rapide. De plus, au fur-et-à-mesure que l'utilisateur utilise le menu, il apprend implicitement la direction de chaque commande. Le mouvement du menu devient un raccourci gestuel exécuté avant-même l'affichage. Parce que la solution du *Marking Menu* exploite l'apprentissage par la répétition, assurant de ce fait une transition fluide du statut de novice à celui d'expert [59, 128, 132], on la tient en général pour la référence en matière de menus gestuels.

Il reste que le *Marking Menu* repose sur une association arbitraire entre l'action à effectuer et la commande : la correspondance doit donc être apprise entièrement, à partir de zéro. De plus, à cause des limitation de l'espace angulaire, ils ne permettent pas en pratique de discriminer plus de 8 directions — et donc plus de 8 commandes (12 lorsque le menu n'est pas hiérarchique) [127]. En conséquence, il est souvent nécessaire de recourir à une organisation hiérarchique, ce qui ajoute des étapes intermédiaire et ralentit la sélection des commandes. Pour augmenter la largeur hiérarchique du menu, certaines variantes utilisent des paramètres supplémentaires comme la courbure ou la forme [17]. La correpondance commande/geste reste toutefois arbitraire (bien qu'une organisation sémantique soit possible).

Les Lettres Augmentées sont un raccourci gestuel combinant une lettre, tracée en une seule fois sans relever le doigt, et un *Marking Menu*. L'idée est de simplifier la mémorisation des commandes et de réduire la charge cognitive en intégrant le nom de la commande au geste. Le tracé se termine par un appendice, pouvant prendre jusqu'à huit directions différentes, ce qui permet de désambiguïser les commandes dont le nom commence par la même lettre.

Ainsi, les Lettres Augmentées sont constituées d'un mnémonique augmenté d'une direction. En supposant que le vocabulaire des commandes du système est connu (par exemple « sauver », « sélectionner », « chercher », « sourire », voir la Figure 5.1), les utilisateurs n'ont besoin d'apprendre que l'appendice de fin de tracé. À la manière

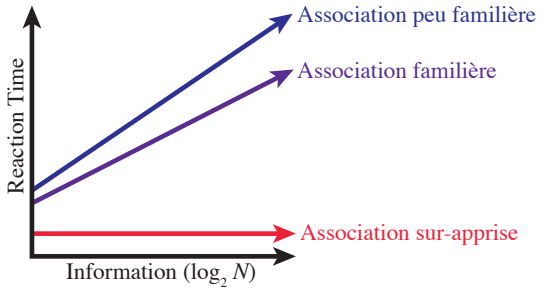


Figure 5.2 – Temps de réaction en fonction de l'information porté par l'ensemble des stimulus [110] et la familiarité avec l'association stimulus-réponse [41]

des *Marking Menu*, les Lettres Augmentées permettent un apprentissage par la répétition et les utilisateurs deviennent peu à peu experts en utilisant le mode novice.

Une propriété intéressante des Lettres Augmentées est qu'elles disposent d'un premier niveau à la fois peu coûteux en terme de recherche et de mémorisation tout en étant particulièrement large : 26 possibilités, une pour chaque lettre de l'alphabet. Ainsi, les Lettres Augmentées exposent une hiérarchie des commandes très plate ce qui, selon les travaux de Scarr et al., constitue une des quatre caractéristiques contribuant à l'obtention de bonnes performances [193, 196]. Par opposition, les *Marking Menus* sont limités par le nombre d'éléments accessibles à chaque niveau (8 au maximum, 12 si le menu n'est pas hiérarchique [128]) ce qui peut provoquer l'introduction de nombreux sous-niveaux si le vocabulaire de commandes est important.

Les Lettres Augmentées ont fait l'objet d'une publication à CHI en 2013 [185].

5.1.1 Sur-apprentissage et automaticité

Les Lettres Augmentées tirent parti du sur-apprentissage de l'ensemble des compétences liées au langage humain. En effet, chez l'adulte lettré, nommer, lire ou écrire sont, après des années de pratique intensive, devenus un automatisme malgré le caractère arbitraire de tout alphabet [192].

À ce propos, la littérature de psychologie classique sur le temps de réaction (ou *Reaction Time* : *RT*) est riche en enseignements. D'après la loi de Hick-Hyman [110], le temps de réaction croît linéairement avec le logarithme du nombre N d'alternatives possibles et que la pente de cette dépendance linéaire dépend de la pratique. À l'extrême, [41] ont montré qu'avec la tâche consistant simplement à lire à haute voix un caractère présenté visuellement, la pente de cette dépendance s'annule. Le temps de réaction n'est alors pas seulement court dans l'absolu, il n'est plus affecté par la taille de l'ensemble des stimuli (voir la Figure 5.2).

Ce type de résultat peut se montrer particulièrement instructif en IHM, no-

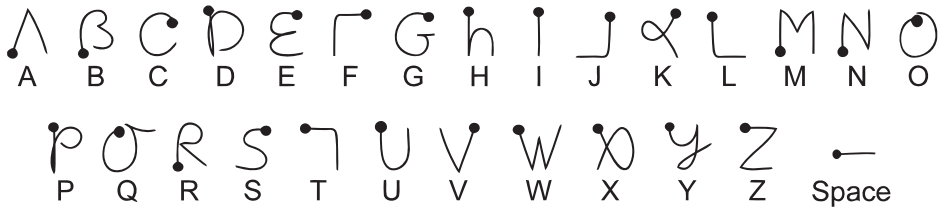


Figure 5.3 – Tracé de lettres Graffiti [55]

tamment lorsqu'il s'agit de définir la manière dont un utilisateur transmet une commande à un système (interaction en entrée). En général, plus le nombre de possibilités augmente, plus le choix de l'action à effectuer prend du temps. Toutefois, si l'association entre action et commande a été pratiquée au point d'être devenue automatique, le nombre d'alternatives n'a plus d'importance. Ainsi, les raccourcis — gestuels en particulier — sont susceptibles de bénéficier grandement des associations sur-apprises, ceci permettant de faire l'économie de l'apprentissage de nouvelles conventions.

Extraire de sa mémoire à long terme le nom d'une commande, identifier l'initiale de son nom, puis esquisser la forme de cette initiale sur un écran tactile devrait demander relativement peu d'effort puisque cette séquence ne comporte aucun élément arbitraire nouveau pour l'utilisateur. Un vocabulaire gestuel s'appuyant sur l'écriture est donc susceptible de grandement faciliter la maîtrise d'un grand ensemble de commandes.

On peut noter que *Gesture Search* est une technique d'interaction dont la finalité est proche : des suggestions de commandes ou d'applications sont proposées en se basant sur un geste et à l'aide d'un algorithme d'apprentissage automatique [140]. Les utilisateur peuvent par exemple atteindre une application en écrivant son nom à main levé, puis la sélectionner dans une liste. Les lettres augmentées permettent de s'affranchir de cette dernière étape.

5.1.2 Description de la technique

Comme expliqué précedemment, une lettre augmentée est une lettre dessinée en un seul tracé à la manière de Graffiti (Figure 5.3) [55]. Cette lettre est « augmentée » avec un appendice (ou « queue ») qui peut prendre jusqu'à 8 orientations différentes (Figure 5.1).

La lettre est (dans le cas idéal) l'initiale du nom de la commande correspondante ce qui permet une association sémantique directe entre le geste et cette commande (voir section 5.1.1). L'appendice permet de différencier les commandes qui commencent par la même lettre et donc d'éviter les collisions de noms.

En théorie, cela permet de définir jusqu'à $26 \times 8 = 208$ commandes différentes. En pratique il faut retirer à ce calcul quelques conflits. En effet, la taille de la queue

(a) « J » avec appendice à gauche

(b) « I » avec appendice à gauche

Figure 5.4 – Exemple de conflit entre Lettres Augmentées

n'étant pas limitée, un « I » avec un appendice orienté vers la gauche est par exemple semblable a un « J » avec un appendice orienté vers la gauche (Figure 5.4). Nous avons comptabilisé six conflits de ce type, ce qui laisse un maximum $208 - 6 = 202$ commandes possibles.

La probabilité qu'une initiale soit partagée par plus de 8 commandes augmente assez lentement avec la taille du vocabulaire [1]. Toutefois, le cas échéant, à la manière d'un *Marking Menu* hiérarchique, nous préconisons d'utiliser une direction spéciale permettant d'accéder à un second niveau contenant les commandes les moins fréquentes.

Deux modes d'utilisation sont possibles avec les Lettres Augmentées.

Mode expert Lorsque l'utilisateur connaît la direction de la commande, il trace la lettre augmentée sans s'arrêter.

Mode novice Dans le cas contraire, il lui suffit de marquer une pause de 500 ms après avoir tracé la lettre pour qu'apparaisse un *Marking Menu* montrant les différentes commandes associées à cette lettre et leurs directions respectives (voir la Figure 5.1). Ce délai a été empiriquement optimisé afin de différencier efficacement une pause marquée volontairement pour ouvrir le menu des immobilisations temporaires liées au mouvement lors d'un angle obtus par exemple.

Si le doigt reste immobile pendant 500 ms après avoir touché la tablette une « antisèche » [132] apparaît dans la partie supérieure de l'écran, indiquant la manière de tracer chaque lettre augmentée. L'antisèche permet à l'utilisateur de découvrir comment utiliser les lettres augmentées lorsqu'il n'y a jamais été confronté (afin de pouvoir être tracé sans relever le doigt, certaines lettres sont légèrement différente de l'écriture manuscrite classique, voir la Figure 5.3).

On remarquera qu'en pratique le mouvement effectué sur la tablette est strictement identique dans les deux modes, novice et expert, et que seule change sa

1. Dans une étude complémentaire, nous avons inventorié le nombre N de commandes accessibles en utilisant les menus de 32 applications sur Mac OS X. En nous référant à ce corpus, la probabilité p qu'une lettre augmentée soit associée à plus de 8 commandes augmente linéairement avec N, l'équation du meilleur ajustement étant $p = 0.0019N - 0.07$, $r^2 = 0{,}92$. Avec 50 commandes, par exemple, la probabilité qu'une initiale soit partagée par plus de 8 commandes est estimée à seulement $p = 0{,}025$.

cinématique : le geste expert, identique à celui du novice, est seulement effectué plus rapidement et sans pause. Ainsi, comme les *Marking Menus* [126, 128], les Lettres Augmentées permettent une transition fluide de l'état de novice à celui d'expert [17].

Il est possible de remplacer les lettres par des symboles représentant non plus l'initiale des commandes, mais des groupes hiérarchiques. Toutefois cette méthode est susceptible de perdre les avantages octroyés par le sur-apprentissage du langage à moins que ces symboles soient bien connus par le type d'utilisateurs auxquels l'application s'adresse.

5.1.3 Implémentation

Dans notre implémentation nous avons eu recours au *$1 algorithm* [233] pour reconnaître les gestes. En mode novice la reconnaissance est effectuée avant l'appendice au moment où l'utilisateur déclenche l'ouverture du menu en marquant une pause : un reconnaisseur spécifique intervient, qui ne reconnaît que les lettres non augmentées. En mode expert, dans lequel l'ensemble du geste (lettre + appendice) doit être reconnu, un système de reconnaissance différent est entraîné pour chaque direction d'appendice possible. Le filtrage selon la direction de l'appendice permet de minimiser le nombre d'erreurs de reconnaissance.

5.1.4 Découvrabilité

L'utilisation du nom des commandes est à la fois une force et une faiblesse de la technique. En effet, si elle bénéficie de l'avantage de l'automatisme cognitif, elle a l'inconvénient d'exiger que l'utilisateur connaisse le nom des commandes. Les commandes ne sont pas rangées par catégories, mais organisées en fonction de leur écriture dans l'antisèche ce qui rend cette technique non optimale pour découvrir les différentes possibilités. Pour cette raison, on doit voir dans les Lettres Augmentées une technique de raccourcis plutôt qu'une technique de menu, un menu étant supposé permettre une exploration hiérarchique des différentes commandes [14]. Nous préconisons donc de les utiliser en tant que tels et, à la manière des raccourcis clavier, d'y adjoindre un menu linéaire traditionnel permettant à un utilisateur novice d'explorer les différentes fonctionnalités disponibles (ce menu pouvant aussi afficher les lettres augmentés adéquates au lieu des accélérateurs clavier, à la manière de Appert et Zhai [7]). Toutefois, contrairement aux raccourcis clavier, il suffit à un utilisateur d'avoir connaissance de l'existence d'une commande (et de son nom) pour pouvoir l'exécuter avec une lettre augmentée.

Plusieurs méthodes peuvent toutefois être mises en place pour supprimer le recours à une autre technique d'interaction et permettre la découvrabilité des fonctionnalités. Tout d'abord, l'anti-sèche montrant la manière dont tracer chaque lettre peut être étendu pour contenir également l'ensemble des commandes. Les travaux de Scarr

et al. [194, 195] indiquent qu'afficher l'ensemble des fonctionnalités en une seule fois peut améliorer les performances des experts tout en maintenant un niveau de performance équivalent pour les novices.

Une autre solution peut être de revoir l'organisation des commandes et d'utiliser les lettres pour désigner non pas des initiales, mais des groupes hiérarchiques Comme mentionné en 5.1.2, il est même possible de remplacer les lettres par un symbole. Toutefois, dans les deux cas, nous perdons l'avantage de l'automatisme du langage.

5.1.5 Expérience

Les Lettres Augmentés peuvent fournir un accès rapide à un grand nombre de commandes. Nous avons mené une expérience inspirée de Bau et Mackay [25] mettant l'accent sur l'apprentissage. L'objectif était d'évaluer les performances des Lettres Augmentées par rapport au *Marking Menus*, une technique largement acceptée comme la référence pour la sélection de commandes gestuelles. Il était demandé aux participants d'apporter un soin particulier à la mémorisation des différents gestes. En conséquence, la principale mesure dépendante était le nombre de commandes gestuelles mémorisées.

Appareillage et participants

Notre expérience a été réalisée à l'aide d'une tablette Archos 80 équipée d'un processeur de 1,5 GHz et fonctionnant sous Android 4. Douze volontaires alors âgés de 21 à 48 ans (5 femmes) ont participé.

Organisation des menus

Étant intéressés par la mémorisation de grands ensembles de commandes, nous voulions être certains de dépasser la mémoire à court terme des utilisateurs Celle-ci est estimée à environ 5-9 *items* [118]. Les *Marking Menu* non hiérarchiques peuvent difficilement dépasser 8 secteurs angulaires (12 au maximum) [133]. Notre *Marking Menu* avait donc un secteur spécial, étiqueté « *others* », qui donnait accès à un second niveau avec 8 commandes supplémentaires. Nous avions donc un total de $7 + 8 = 15$ éléments accessibles avec le *Marking Menu*. Nous aurions préféré un plus grand nombre de commandes, mais cela aurait désavantagé les *Marking Menus* face aux Lettres Augmentées. Les commandes furent choisies de manière à former 5 groupes de commandes commençant par la même lettre. Cinq lettres différentes furent donc utilisées, trois avec 4 directions d'appendices possibles et deux avec 2 Il y avait donc au total $3 \times 4 + 2 \times 2 = 16$ éléments possibles dans le cas des Lettres Augmentées.

Pour chacun des deux menus, les participants ont été testés sur 12 commandes différentes parmi l'ensemble des commandes possibles. Pour les *Marking Menus*, deux commandes étaient inutilisées sur le second niveau et une sur le premier. Pour

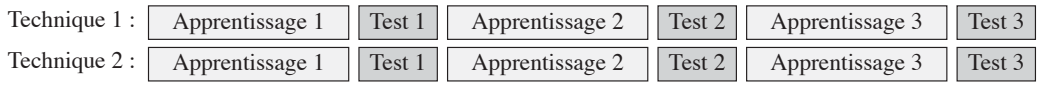


Figure 5.5 – Organisation des blocs de l'expérience

les *Lettres Augmentées*, deux commandes étaient inutilisées sur chacune des lettres comportant 4 éléments.

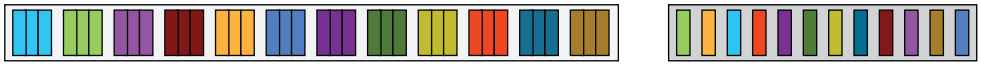
Méthode

Nous avons eu recours à un jeu d'items (nous utiliserons l'expression jeu de commandes) contenant des noms de pays et à un autre jeu contenant des noms d'animaux. Afin d'éviter toute interférence due à l'apprentissage, nous n'avons pas mélangé les deux techniques : l'expérience était divisée en deux parties successives, chacune dédiée à une technique. L'ordre des techniques ainsi que le jeu de commande associé à chaque technique étaient balancés entre participants à l'aide d'un carré latin.

Tâche et stimulus Les participants devaient essayer de mémoriser autant de gestes que possible. Chaque bloc commençait avec l'affichage du nom de la technique concernée, l'affichage précisant également s'il s'agissait d'un bloc d'apprentissage ou de test. Chaque essai était précédé par un compte à rebours circulaire. À la fin de ce compte à rebours, l'essai se déclenchait. Le nom de la cible apparaissait alors dans la partie supérieure de l'écran et le participant devait effectuer le geste correspondant. Un essai prenait fin lorsque l'utilisateur relâchait le doigt de la tablette.

Blocs d'apprentissage et blocs de test Notre protocole est schématisé dans les figures 5.5 et 5.6. Les mêmes 12 éléments étaient présentés au sein de trois blocs d'apprentissage successifs. Pendant les blocs d'apprentissages, les participants étaient libres d'utiliser le menu des techniques ou bien d'exécuter le geste en mode expert, sans le secours de l'antisèche. Durant ces blocs, chaque élément était demandé 3 fois de suite (Figure 5.6a). Ces répétitions avait pour objectif d'accélérer la mémorisation du geste (elles étaient rarement exécutées avec l'aide du menu, le mouvement étant encore en mémoire à court terme). À chaque bloc d'apprentissage succédait un bloc de test où le menu était indisponible et où les participants devaient donc se remémorer le geste.

L'expérience a duré environ 45 minutes, avec $2 \times 3 \times (3 \times 12 + 12) = 288$ essais mesurés par participant, soit un total de $288 \times 12 = 3\,456$ essais.



(a) Exemple de bloc d'apprentissage (b) Exemple de bloc de test

Figure 5.6 – Organisation des essais de l'expérience

5.1.6 Résultats et discussion

Performances du mode expert

Une ANOVA indique un effet significatif du facteur Technique sur le taux de mémorisation ($F_{3,11} = 7{,}08$, $p < 0{,}01$). Ce taux de mémorisation (figure 5.7) a été en moyenne plus élevé de 24,5 % avec les Lettres Augmentées (76,6 %, soit 9,2 commandes mémorisées) que pour les *Marking Menus* (52,1 %, 6,25 commandes mémorisées).

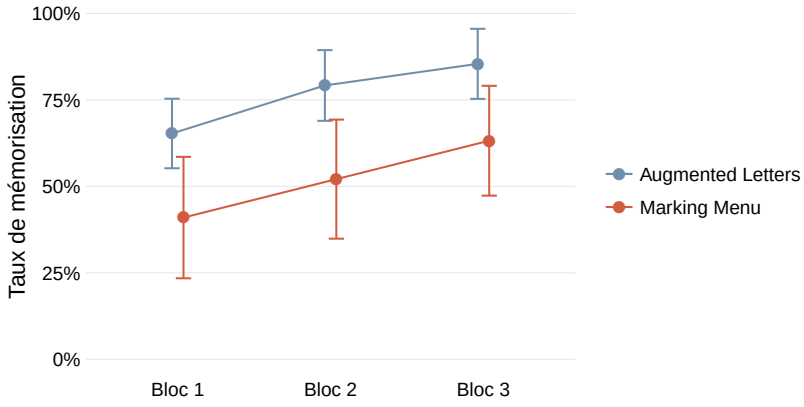
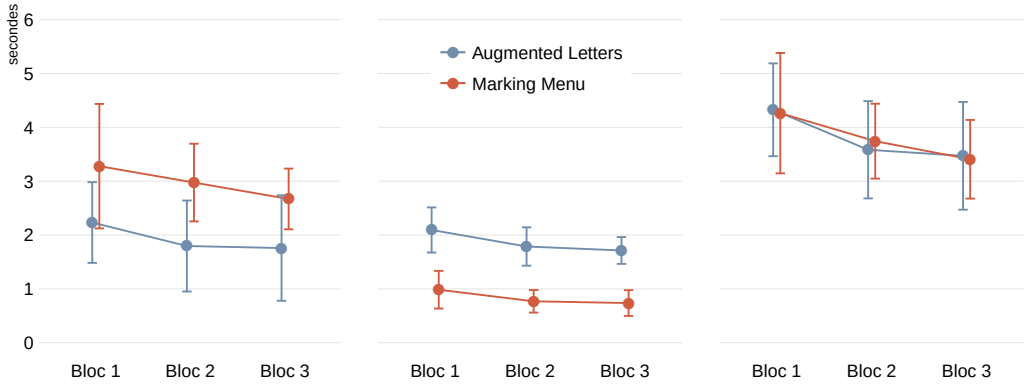


Figure 5.7 – Taux de mémorisation des Lettres Augmentées et des *Marking Menus*

Les vitesses moyennes mesurées sont présentées sur la Figure 5.8. Le temps d'exécution en mode expert est sensiblement le même pour les Lettres Augmentées que pour les *Marking Menus* (3,80 sec contre 3,79 sec). Toutefois, une analyse plus fine a mis en évidence une différence dans la répartition de cette durée entre le temps de réaction et le temps d'exécution du geste. Il a fallu plus de temps aux participants pour dessiner une lettre augmentée que pour tracer le geste d'un *Marking Menu* (1,86 sec contre 0,83 sec, $F_{1,11} = 178$, $p < 0{,}01$), mais cette différence est entièrement compensée par un temps de réaction plus court (1,92 sec contre 2,97 sec, $F_{1,11} = 19$, $p < 0{,}01$). Ce dernier constat est conforme à l'idée que le temps d'ouverture d'une lettre augmentée est favorisé par la forte familiarité de l'écriture d'un mot, que ce soit le nom d'un pays ou d'une commande informatique.



(a) Temps de réaction (b) Durée du geste (c) Temps de sélection total

Figure 5.8 – Performances en vitesse des Lettres Augmentées et des *Marking Menus*

Performances du mode novice

Nous avons également observé la manière dont les participants utilisent chaque technique lorsqu'ils souhaitent accéder à une commande pour la première fois. Pour cela, nous avons examiné les temps d'exécution du mode novice lors du premier bloc d'apprentissage de l'expérience. Nous avons mesuré un temps de sélection de 6,6 sec en moyenne pour les Lettres Augmentées et de 7,9 sec en moyenne pour les *Marking Menus* (un test de Student donne $t = 1{,}46$, $p = .09$, test unilatéral). Bien que statistiquement peu significatif, ce résultat semble toutefois à nouveau concorder avec l'hypothèse qu'un codage sur-appris induit un avantage immédiat.

Utilisation spontanée du mode expert

Durant les blocs d'apprentissage, les participants étaient libres de marquer une pause pour que le menu s'affiche (mode novice) ou d'effectuer le geste directement (mode expert). En moyenne, sur les deux dernières phases d'apprentissage de l'expérience, l'utilisation spontanée du mode expert fut de 22,2 % plus fréquente avec les Lettres Augmentées (66,0 %) qu'avec les *Marking Menu* (43,8 %, $t = 2{,}14$, $p = 0{,}03$, test unilatéral). Nous en concluons qu'il était plus facile pour nos participants d'apprendre l'appendice d'une lettre augmentée que les directions des premier et second niveaux des *Marking Menus*.

5.1.7 Conclusion sur les Lettres Augmentées

Pour conclure, les Lettres Augmentées sont une technique d'interaction qui permet d'accéder à un très grand nombre d'éléments tout en étant faciles à mémoriser, grâce à l'exploitation du langage. Bien qu'il soit fort probable que les commandes les plus fréquentes, après un entraînement suffisamment conséquent, s'exécutent

plus rapidement avec un *Marking Menu* en raison du geste plus court qui leur est associé, notre expérience semble indiquer que sur l'ensemble des commandes, les Lettres Augmentées ne sont pas en reste en terme de vitesse. Notre position sur ce point est que le niveau d'expertise requis pour obtenir des performances supérieures avec les *Marking Menu* est tel qu'il ne sera probablement que rarement atteint. Notons également que cette expérience s'est limitée à une base de 12 éléments, un nombre assez faible de commandes, afin de ne pas trop désavantager les *Marking Menu*. Or, si nous nous référons à la loi de Hick-Hyman [110], plus grand est le vocabulaire de l'application, plus important est l'avantage apporté par l'exploitation du langage.

5.2 Barre de Menus *Multitouch*

Figure 5.9 – Barre de Menus *Multitouch*. La barre de menu est affichée en bas à gauche. Le menu s'ouvre dans la partie supérieure de l'écran.

Un des freins à l'adoption des techniques d'interaction gestuelles, comme les Lettres Augmentées, est le phénomène de qwertynomie [65, 242] : les utilisateurs, peu familiers avec ce type d'interaction, sont attachés aux modalités auxquelles ils sont habitués malgré le potentiel gain en performance que permettrait une transition.

Pour contrer ce phénomène, une solution — peu explorée dans la littérature — peut être la conception de techniques *rétrocompatibles* avec les interactions traditionnelles, par exemple les barres d'outils et les barres de menus pour la sélection de commandes sur tablette tactile. Cette approche sensiblement différente de celle utilisée avec les Lettres Augmentées est celle que nous avons choisi d'explorer lors de la conception de la Barre de Menus *Multitouch*. L'idée est de doter les barres de menus traditionnelles sur tablette tactile d'une méthode de sélection (plus) rapide destinée en particulier (mais pas nécessairement exclusivement) aux utilisateurs experts.

Ainsi, la Barre de Menus *Multitouch* (Figure 5.9) se présente comme une barre de menus classiques dans laquelle l'utilisateur sélectionne la commande en tapant le bouton ouvrant le menu où elle se situe, puis, une fois le menu ouvert, tape l'item représentant la commande cible. Toutefois, lors de cette dernière étape, la commande est sélectionnée sans pointer, en touchant la surface de la tablette n'importe où mais avec le nombre de doigts n_d correspondant au numéro de la commande cible, c'est à dire à sa position au sein du menu. Cette façon d'exploiter le nombre de doigts s'inspire ainsi de la technique [16, 21].

La Barre de Menus *Multitouch* a été implémenté en HTML, SVG et JavaScript. Une démonstration en ligne est accessible à l'adresse `http://quentinroy.fr/demos/cube?menu=mttb`. Elle a été testée sur Safari Mobile 8.0, Safari 8.0, Mozilla Firefox 38.0 et Google Chrome 43.0.

5.2.1 Rétrocompatibilité et règles de conception

Nous définissons la technique T_1 comme *rétrocompatible* avec la technique T_0 si T_1 inclue l'ensemble des comportements propres à T_0. En d'autres termes, T_1 peut être utilisée exactement comme s'il s'agissait de T_0.

Le principe de *rétrocompatibilité* est inspiré des interfaces logicielles (*APIs*). L'interface d'un composant logiciel définit la manière dont il doit être utilisé au sein d'une application. D'une version à une autre, il est possible d'ajouter sans grand risque de nouveaux éléments à cette interface, mais il est dangereux de modifier l'existant sous peine de briser la rétrocompatibilité avec les versions précédentes. Parfois, malheureusement, la contrainte de rétrocompatibilité fait obstacle à l'évolution d'un composant logiciel, le programmeur pouvant être amené à enfreindre cette règle. En conséquence, les utilisateurs du composant (des programmeurs) sont généralement forcés de récrire une partie de leur application afin de l'adapter aux changements. Ceci implique une charge de travail parfois très conséquente ce qui peut les amener à renoncer à la transition (c'est-à-dire à conserver la version antérieure) voire même à abandonner la bibliothèque, éventuellement au profit d'une autre plus stable.

En ce qui concerne les interfaces utilisateurs, David a ainsi mis en évidence le phénomène dit de la *qwertynomie* et montré qu'une fois qu'un utilisateur s'est

habitué à une solution, même sous-optimale, il est extrêmement difficile d'en instaurer une nouvelle quelle que soit le gain de performance : c'est un fait que les claviers Dvorak, de 20 à 40% plus performants que les claviers Qwerty, ne les ont jamais supplantés [65]. Zhai et al. ont récemment fait valoir à quel point ce mécanisme handicape la recherche sur les interfaces gestuelles [242]. La majorité des propositions de la littérature dans le domaine de l'interaction gestuelle semblent peu soucieuses de cette problématique, les auteurs n'hésitant pas à proposer aux utilisateurs des méthodes d'interaction significativement différente de l'existant. A nos yeux la rétrocompatibilité des interfaces utilisateurs est un principe dont l'importance tend généralement à être sous-estimée en IHM.

Une technique d'interaction rétrocompatible est donc une technique qui prend pour base une technique existante : un utilisateur qui ne souhaite pas profiter des nouvelles fonctionnalités a la possibilité de les ignorer entièrement et de continuer à manipuler l'interface comme d'habitude. Aucun apprentissage à priori n'est donc obligatoire. Toutefois la technique intègre également une amélioration permettant soit de gagner en productivité dans les actions déjà réalisables soit d'exécuter des actions auparavant non disponibles. On peut noter qu'il demeure tout de même un risque que les utilisateurs n'apprennent jamais les nouvelles interactions ainsi proposées.

Notre hypothèse est que les techniques d'interaction rétrocompatibles avec les techniques traditionnelles ou habituelles peuvent

1. réduire ou supprimer le phénomène de qwertynomie,
2. aider à la transition progressive vers de nouvelles modalités d'interaction (par exemple à la transition des techniques d'interaction basées sur le pointage vers une solution gestuelle) et
3. remplacer facilement une technique d'interaction au sein d'une application existante sans requérir une révision profonde de l'ensemble de l'ergonomie du système.

Le défi de la conception de ce type de techniques réside essentiellement en 3 points :

1. permettre un accroissement des performances viables, ou supporter un vocabulaire de commandes plus important,
2. tout en s'intégrant suffisamment à la technique d'origine afin de réduire au maximum le coût d'un éventuel changement de modalité [59, 196],
3. et en minimisant les modifications apportées à la manière traditionnelle d'interagir avec la technique afin de garantir la meilleure rétrocompatibilité possible.

Ainsi, la Barre de Menus *multitouch* que nous proposons reste entièrement compatible avec la manière dont les utilisateurs interagissent avec une barre de menus traditionnelle.

5.2.2 Stratégies d'utilisation de la barre de Menu *Multitouch*

La Barre de Menus *Multitouch*, bien que très simple dans son principe, s'avère assez flexible, l'utilisateur final ayant le choix entre plusieurs stratégies.

Méthode traditionnelle

L'utilisateur ouvre le menu en tapant sur un bouton de la barre de menu. Le sous-menu apparaît, dans lequel l'utilisateur fait sa sélection en tapant de nouveau sur un bouton.

Sélection *multitouch* en deux étapes

L'utilisateur tape sur le bouton du menu cible dans la barre de menus. Le sous-menu étant apparu, l'utilisateur appuie avec n_d doigts, n'importe où sur la tablette à l'exception du sous-menu et de la barre de menus. L'élément de rang n_d du menu est présélectionné et mis en relief. L'utilisateur peut alors ajouter ou retirer des doigts de la surface, l'élément présélectionné étant mis à jour en conséquence. La sélection est validée lorsque l'utilisateur relève tous les doigts de la surface.

L'ensemble des doigts n'étant jamais relevés exactement en même temps, un délai de 150 ms est appliqué avant la mise à jour de la présélection lorsqu'on retire un doigt de la surface afin d'éviter que celle-ci ne soit involontairement modifiée au moment de la validation.

Il est à noter que le nombre d'éléments sélectionnables avec cette méthode au sein d'un menu donné est évidemment limité au nombre de doigts. S'il contient davantage de commandes, celles-ci restent cependant accessibles avec la méthode traditionnelle.

Sélection *multitouch* en une étape

Il est de plus possible de fusionner l'ouverture du menu avec la sélection de la commande.

L'utilisateur touche alors la surface avec n_d doigts en posant l'un d'entre eux sur le bouton du menu cible dans la barre de menus (voir la Figure 5.10). La commande de rang $(n_d - 1)$ est présélectionnée et mise en relief. L'utilisateur peut alors ajouter ou supprimer des doigts et l'élément présélectionné est mis à jour en conséquence. L'utilisateur retire ses doigts pour confirmer la sélection.

Dans cette variante, la question du numéro de l'élément sélectionné mérite discussion. Dans les premières versions de la Barre de Menus *Multitouch*, poser n_d doigts sur la surface dont un sur le bouton d'ouverture du menu permettait de sélectionner l'élément de rang n_d et non pas celui de rang $n_d - 1$ comme c'est le cas dans la dernière itération de la technique. Cette méthode ne permettait toutefois pas à la technique d'être rétrocompatible avec les barres de menu traditionnelles :

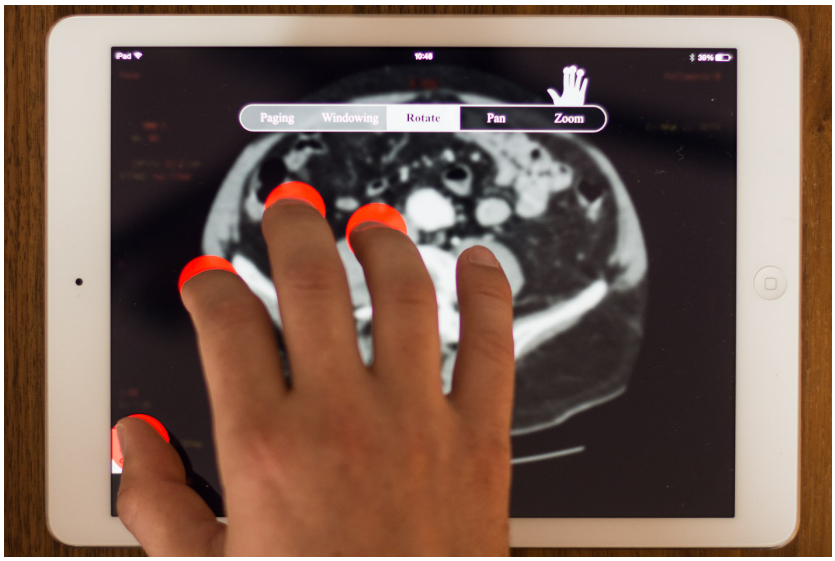


Figure 5.10 – Sélection en une étape avec la Barre de Menus *Multitouch*

taper avec un doigt sur le bouton du menu servait à sélectionner le premier élément de ce menu et non plus à l'ouvrir. Nous avons donc décidé de ne pas tenir compte de ce doigt pour la sélection de la commande. Nos tests informels ont montré que le modèle mental des utilisateurs s'accommodait bien de ce mécanisme. Une conséquence de ce choix est que seuls 4 raccourcis sont alors disponibles (contre 5 auparavant).

D'autres alternatives pourraient être considérées, comme par exemple effectuer un appui long ou un déplacement du doigt pour ouvrir le menu, mais elles poseraient également le problème de la rétrocompatibilité. Inversement, un accès direct à une fonctionnalité pourrait nécessiter un appui suivi d'un déplacement des doigts mais ceci ralentirait légèrement l'interaction (et interférerait dans le cas d'une opération impliquant du contrôle continu, comme exposé plus bas).

Sélection à deux mains

À l'instar de Bipad [222], il est possible d'utiliser cette technique avec les deux mains : L'utilisateur presse le bouton de la barre de menu avec une main et utilise l'autre pour sélectionner la commande à l'aide de la méthode de son choix.

Sélections multiples

En maintenant appuyé le bouton du menu, il est possible d'effectuer plusieurs sélections sans le refermer.

L'utilisateur touche le bouton du menu cible dans la barre de menus et le maintient enfoncé. Le sous-menu étant apparu, l'utilisateur tap avec n_d de ses doigts restant (ou avec son autre main) tout en gardant le bouton du menu enfoncé. L'élément de rang n_d du menu est sélectionné. Il peut répéter cette opération tant qu'il maintient le bouton du menu enfoncé.

Fusion de la sélection de la commande et du contrôle continu

Enfin, la Barre de Menus *Multitouch* permet également de fusionner la sélection de la commande avec le contrôle continu à la manière des *Control Menus* [169], des *FlowMenus* [90], des *FingerMenus* (Partie 4.2) et des *SigmaMenus* (Partie 4.2) [57, 88]. Une fois la commande présélectionnée, commencer à faire glisser le(s) doigt(s) permet de valider automatiquement la sélection et commencer à manipuler le contrôle associé dès qu'une distance seuil a été parcourue. La barre d'outils se comporte alors comme le *Multi-tap Slider* de Damaraju et al. [64] ou le *Finger-Count* de Bailly et al. [21]. Toutefois, à la différence des techniques précédentes, une fois la sélection validée, il n'est plus nécessaire de maintenir le nombre de doigts associé à la sélection de la commande. Ceux qui sont superflus peuvent donc être relâchés pour gagner en précision et en confort.

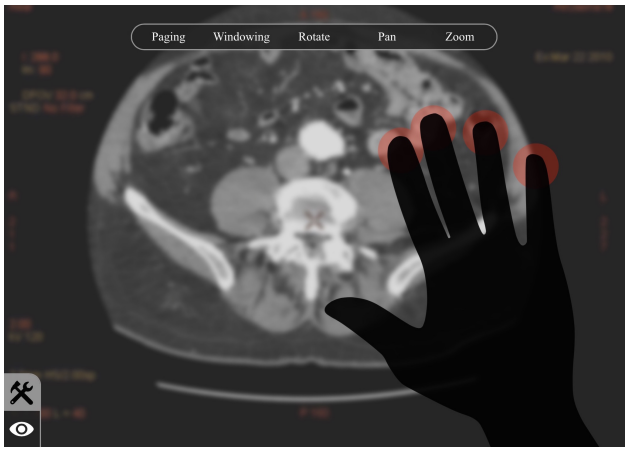
5.2.3 Choix de conception

Positionnement du menu

Afin de laisser la place à la sélection *multitouch*, les sous-menus apparaissent tous dans la partie supérieure de l'écran au lieu d'apparaître à côté du bouton du menu (voir la Figure 5.9). Cette méthode a le défaut d'allonger le temps de pointage lorsqu'un utilisateur sélectionne une commande en tapant dans le menu. On peut toutefois arguer qu'un corollaire est d'inciter davantage les utilisateurs à recourir à la sélection *multitouch* qui est plus efficiente (la tactique consistant à rendre plus difficile l'accès au mode novice, de manière à encourager l'adoption du mode expert, a déjà été utilisée dans la littérature [3]). Il est toujours possible de placer le menu à une position plus traditionnelle, mais on s'expose à plus d'erreurs de sélection car il serait alors possible d'activer un bouton involontairement en utilisant la stratégie de sélection *multitouch*.

Découvrabilité de l'interaction *multitouch*

L'une des difficultés à régler dans la mise au point de la Barre de Menus *Multitouch* est de savoir comment notifier aux utilisateurs la présence d'une autre manière d'interagir, celle reposant sur le nombre des doigts. Une première solution a été de mettre en place une animation de la main déclenchée par l'ouverture du menu (Figure 5.11a). Toutefois, nos tests informels ont montré que les utilisateurs, dont le



(a) *Feedforward* placée à droite de l'écran

(b) *Feedforward* placée juste au dessus du menu (méthode sélectionnée)

Figure 5.11 – Étude du retour utilisateur sur la disponibilité de l'interaction *multitouch* de la Barre de Menus *Multitouch*

regard est alors fixé sur le menu juste ouvert, avaient tendance à ne pas remarquer l'animation. Nous l'avons alors déplacée juste au-dessus du menu (Figure 5.11b). Bien que nécessairement plus petite, elle se trouve alors à un emplacement exposé à l'attention de l'utilisateur. Lors de la sélection d'une commande avec la méthode traditionnelle, cette main affiche la manière dont cette même commande peut être sélectionnée à l'aide de la méthode *multitouch* (voir Figure 5.11b).

5.2.4 Conclusion sur la Barre de Menus *Multitouch*

Nos tests informels sur la Barre de Menus *Multitouch* sont encourageants. Nos participants se sont montrés particulièrement enthousiastes et ont qualifié la Barre de Menus *Multitouch* comme « efficace », « confortable » et « évidente ». Nous pensons que ces retours positifs sont au moins en partie le résultat de notre souci de rétrocompatibilité avec les barres de menus traditionnelles.

La Barre de Menus *Multitouch* a été retenue pour notre prototype de radiologie

et est susceptible d'être déployée au sein de la future application commerciale de GE HealthCare. décidé de ne pas y donner de suite pour l'instant par crainte de complications juridiques du fait de la position de force d'Apple en ce qui concerne les techniques d'interaction *multitouch*. Une publication défensive, leur permettant de se prémunir d'une éventuelle attaque, a toutefois été déposée.

Une validation formelle reste à mettre en place et nous réfléchissons à la manière dont le principe de rétrocompatibilité peut être évalué.

6 Identification des doigts et le canal verre+peau

La conception de nouvelles techniques d'interaction — comme la Barre d'Outils *Multitouch*, les Lettres Augmentées, le *Finger Menu* ou encore le *Sigma Menu* présentées dans les chapitres précédents — permet d'améliorer l'interaction avec un système en tirant plus avantageusement parti des dispositifs d'entrée disponibles. Une approche différente est d'identifier de nouveaux types d'entrée pouvant être avantageusement mis à profit lors de l'interaction.

Le *FingerMenu*, présenté en 4.1, permet d'atteindre des performances en sélection de commandes intéressantes. Pour ce faire, il repose sur un geste d'activation qui consiste à poser les 5 doigts de la main sur la tablette. Ceci permet d'identifier chaque doigt et d'associer à chacun une commande. Le *Finger Menu* laisse donc penser que l'identification des doigts peut être un canal intéressant pour les tablettes tactiles. Comme le montrent les travaux de Holz et Baudisch ou de Goguey et al., cette approche reçoit de plus en plus d'attention dans la littérature [10, 32, 69, 82, 83, 106, 139, 147, 223, 226].

La reconnaissance de l'identité des doigts offre plusieurs avantages. Tout d'abord, elle permet d'augmenter le vocabulaire en entrée tout en maintenant partiellement la rétrocompatibilité avec le style d'interaction traditionnel : par exemple, le même bouton peut permettre d'invoquer différentes commandes en fonction du doigt utilisé. Il convient de noter que sur les tablettes tactiles, mais aussi sur les *smartphones*, l'interaction est généralement davantage limitée par l'entrée (la surface tactile) que par la sortie (l'affichage). En effet, grâce à la haute résolution des écrans modernes, les icônes — et même souvent le texte — restent reconnaissables bien après avoir atteint une taille trop petite pour l'extrémité d'un doigt. Plusieurs icônes peuvent donc être être affichées sur le même bouton et l'identification des doigts peut être exploitée pour atteindre la commande associée à chacune d'elles.

Cette stratégie peut être exploitée de deux manières différentes : (1) soit en accroissant le nombre total de commandes disponibles, (2) soit en réduisant le nombre de boutons tout en conservant un accès direct (c'est-à-dire un accès ne nécessitant pas l'exploration d'un menu, d'une liste déroulante, etc). Diminuer la quantité de boutons peut permettre soit de minimiser la surface d'affichage mobilisée pour la sélection des commandes, soit d'agrandir la taille des boutons et donc d'améliorer les performances de pointage (ceci ayant pour effet de réduire

l'indice de difficulté de la loi de Fitts [70]).

L'identification des doigts peut également servir à fournir des raccourcis pour invoquer des commandes fréquentes ou favorites. On peut par exemple imaginer « sélectionner » un objet en le touchant avec l'index, le « copier » en le touchant avec le majeur et le « coller » en touchant l'écran avec l'annulaire.

Nous appelons :

Canal verre+peau le canal mixte qui, lors du contact avec une surface tactile (généralement en verre pour une tablette tactile), tient compte non seulement des coordonnées XY sur cette surface, mais également des coordonnés sur la surface du corps de l'utilisateur l'utilisateur (plus specifiquement, dans notre cas, de l'identité du doigt responsable de ce contact).

Canal verre le canal tactile traditionnel qui tient compte uniquement des coordonnées XY sur cette surface.

Alors que la plupart des travaux se sont intéressés à la faisabilité technologique de l'identification des doigts [10, 15, 82, 106, 223, 226], nous nous sommes concentrés sur la validation de cette approche : le canal mixte verre+peau permet-il un gain de productivité sur tablette tactile ? Dans quelles conditions ?

Pour répondre à ces questions, nous avons mené une étude utilisateur comparant les performances du canal mixte verre+peau et du canal tactile traditionnel, en faisant varier le nombre de commandes disponibles. Pour une surface de pointage donnée, nos résultats montrèrent que si le canal traditionnel est efficient avec peu de commandes, il est rapidement dépassé par le canal mixte verre+peau lorsque le nombre de commandes augmente. De plus, le taux d'erreurs du canal mixte verre+peau augmente à un rythme fortement réduit quand la taille du vocabulaire s'accroît.

En nous reposant sur la théorie de l'information de Shannon [200], nous avons également constaté que la bande passante maximum (ou, plus précisément, le niveau maximal de débit possible) est considérablement plus élevé pour le canal mixte verre+peau, et que ce maximum est atteint pour un nombre de commandes sensiblement plus grand. Ceci montre que les utilisateurs sont capables de manipuler des vocabulaires plus importants à l'aide du canal mixte verre+peau. La théorie de l'information de Shannon [200] nous a également permis de déterminer précisément le point où, pour une surface de pointage donnée, les performances du canal mixte verre+peau commencent à surpasser les performances du canal verre.

Ces travaux ont été publiés et présentés à la conférence internationale Interact en septembre 2015 [184].

6.1 Augmenter l'expressivité des surfaces tactiles

Face à la petite taille des écrans mobiles et au problème du gros doigt [33, 171, 182, 220], plusieurs modalités ont été proposées dans le but d'augmenter l'expressivité

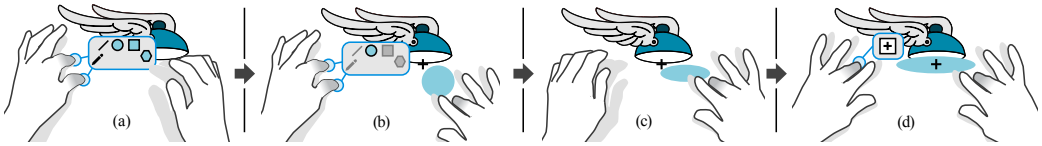


Figure 6.1 – Adoiraccourcix [82]. (a) un accord de doigts de la main non dominante donne un *feedforward* montrant les différentes commandes associées aux doigts de la main dominante. (b) Le majeur de la main dominante sélectionne l'outil de création d'ellipses. (c) Le mouvement de la main dominante contrôle les paramètres de la commande. (d) Un autre accord de la main non dominante agit comme modificateur sur le mouvement de la main dominante.

de la saisie tactile.

La plus répandue est certainement le *multitouch* [137] qui a été popularisé par les téléphones Apple. Une exploitation particulière du *multitouch* est le *Finger-Count* [16, 21], qui permet de sélectionner une commande en se fondant sur le nombre de contacts avec la surface tactile. D'autres modalités ont également été proposées comme la combinaison du mouvement et du tactile [103] ou encore la pression [33, 97, 153, 173, 178]. La bande passante de cette dernière est toutefois faible en raison d'un temps de sélection élevé (de l'ordre de 2 sec en l'absence de retour visuel) et de la difficulté à différencier plus que 5 à 7 valeurs différentes [153]. Wang et al. tirèrent parti de l'orientation du doigt, notamment pour contrôler un ou plusieurs paramètres [225]. Enfin, *TapSense* distingue la frappe de différente partie du doigt (plat, pointe, ongle, articulation) à partir de leur signature acoustique [95].

À la suite de ces travaux, l'identification du doigt — qui permet de prendre en compte conjointement les coordonnées d'un contact sur l'écran et sur la peau — semble prometteuse [10, 32, 69, 82, 83, 106, 139, 147, 223, 226].

De nombreuses études se sont concentrées sur l'exécution de commandes ou d'actions en fonction du doigt [10, 32, 139, 147, 223]. Par exemple, Adoiraccourcix associe différents modificateurs aux doigts de la main non dominante et différentes commandes aux doigts de la main dominante [82] (Figure 6.1). D'autres travaux identifient le doigt lors de la pression d'un bouton physique [208, 226], une solution maintenant intégrée à certains *smartphones* chez Apple, Samsung ou encore HTC. L'avantage de la technique est que l'identification peut être effectuée même si le bouton n'est pas enfoncé. Une variante *multitouch* des *Marking Menus* [126] qui fait la différence entre les différents doigts a également été étudiée par Lepinski et al. [139].

Certains chercheurs se sont intéressés à la la découvrabilité des commandes dépendantes des doigts. Par exemple, Sugiura et Koseki [208] identifièrent le doigt dès que l'utilisateur touche un bouton (physique). Ils utilisèrent cette propriété

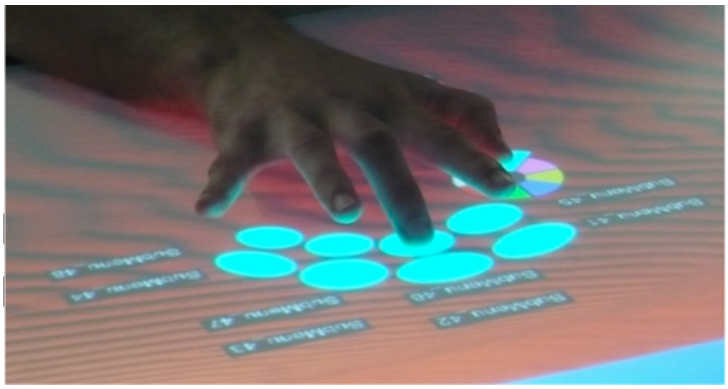


Figure 6.2 – *Multitouch Menu* (*MTM*) [15]

pour afficher un retour visuel sur le nom de la commande correspondante avant que le bouton ne soit pressé (la pression du bouton valide la sélection). Ceci n'est cependant pas compatible avec la plupart des systèmes tactiles, qui, le plus souvent, ne disposent que d'un seul état actif [44] et ne reconnaissent pas la pression. Au et Tai [10], à la manière des *FingerMenu* présentés précédemment (Partie 4.1), proposent un menu listant une commande sous chaque doigt. Pour invoquer ce menu, un utilisateur doit toutefois auparavant poser les cinq doigts de sa main ce qui permet de localiser chacun des doigts.

6.2 Technologie d'identification des doigts

De nombreuses méthodes ont été mises en place pour permettre de différencier les doigts les uns des autres lors du contact avec une surface.

Dans certaines circonstances, les doigts peuvent être identifiés à partir des dispositifs d'entrée traditionnels propres aux tablettes tactiles. En général, ceci consiste à tirer parti d'un geste particulier de la main. En supposant une posture de main détendue, l'utilisateur doit toucher la surface d'une certaine manière, par exemple Wagner et al. reconnait des combinaisons non ambiguës de doigts [223], ou exécuter une séquence temporelle spécifique [10]. D'autres techniques telles que *MTM* [15] ou Arpège [79] ne permettent pas d'identifier directement les doigts, mais font des hypothèses sur leurs coordonnées, notamment en s'appuyant sur la position déjà connue d'autres parties du corps. Par exemple, *MTM* utilise la localisation du talon de la main (Figure 6.2).

Une autre méthode consiste à équiper les utilisateurs de gants numériques [207] ou autres dispositifs portés permettant le suivi des doigts. Par exemple, Goguey et al. attachèrent des *GameTraks*[1] aux doigts de l'utilisateur (Figure 6.3) [82, 83].

1. *GameTrak* est un contrôleur de jeu conçu pour la PlayStation 2 de Sony. Il est équipé

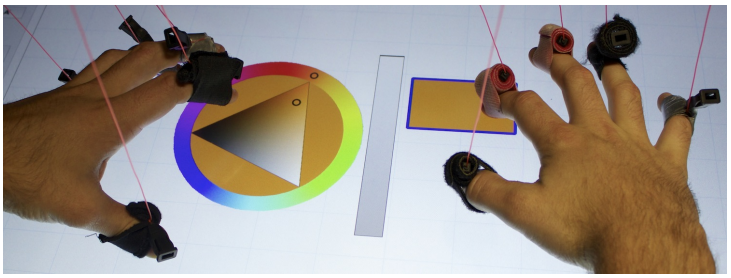


Figure 6.3 – Adoiraccourcix [82, 83]

L'inconvénient de ce type de techniques est qu'elles nécessitent une instrumentation de l'utilisateur et/ou une phase d'étalonnage.

Les techniques de vision par ordinateurs peuvent servir à identifier les doigts sans nécessiter de gestes particuliers. La caméra est parfois située derrière la surface interactive comme avec les tables *multitouch* FTIR (c'est la méthode utilisée par Lepinski et al. [139]) ou être placée au-dessus (c'est la méthode utilisée par Benko et al. [31]). Le principe est de comparer le bout des doigts (obtenu par vision par ordinateur) avec la position du ou des points de contact sur la surface tactile. Similairement, certains systèmes commerciaux sont en mesure de suivre le mouvement de chacun des doigts dans l'espace 3D (par exemple *Microsoft Kinect* et *Leap Motion*). Kung et al. [124], en particulier, utilisèrent ce type de produit afin de mettre place un dispositif d'entrée capable d'identifier les doigts lors du contact avec une surface. Les techniques de vision par ordinateur peuvent toutefois échouer pour certaines postures de la main [69]. Recourir à des marqueurs de couleur [83, 226] ou des tags fiduciaires [147] permet, dans une certaine mesure, de réduire ce problème. Ceci est toutefois difficile à mettre en place en situation réelle, car ces techniques nécessitent d'instrumenter les utilisateurs et restent donc généralement confinées aux laboratoires de recherche.

Des capteurs intégrés à la surface tactile ont également été mis à profit. Sugiura et Koseki [208], par exemple, se sont servir d'un scanner d'empreintes digitales. Leur système permet de déclencher une commande en fonction d'un doigt, mais pas de suivre la position du ou des points de contact. Holz et Baudisch confectionnèrent un *touchpad* pouvant lire les empreintes digitales [107], puis, plus récemment, une table tactile [106].

Une dernière approche consiste à analyser les courants électriques accompagnant l'activité musculaire (EMG), en particulier au niveau de l'avant-bras, afin de reconnaître la signature caractéristique de l'utilisation d'un doigt en particulier [32].

---

de deux cordelettes rétractables dont l'extrémité est munie d'un attache-membre. L'appareil est capable de suivre la position 3D de ce qui y est fixé.

6.3 Verre+peau : une nouvelle classe d'interaction

Plusieurs interacteurs, tels que les barres d'outils ou les menus par exemple, s'appuient exclusivement sur l'arrangement spatial des boutons sur l'écran. Au cours de l'interaction avec ces *widgets*, le système exploite uniquement les coordonnées du point de contact entre le doigt et l'écran sur la surface tactile. Dans cette section, nous montrons comment l'identification des doigts peut offrir des propriétés intéressantes pour améliorer la sélection de commandes sur écrans tactiles. Nous explorons différents exemples de techniques tirant parti du canal mixte verre+peau et pouvant être mise en œuvre par un concepteur afin d'améliorer l'interaction avec ses applications.

6.3.1 Boutons multifonctions

Accroître le vocabulaire en entrée

L'écran principal de l'iPhone peut fournir un accès direct de 20 à 24 applications (un arrangement de 4×5 ou 4×6 boutons, selon le modèle). Avec la distinction de cinq doigts différents, il est possible de multiplier par 5 le nombre d'applications accessibles sur chaque écran.

Réduire le nombre de boutons

Peut-être plus intéressant, le canal mixte verre+peau peut également permettre de réduire le nombre de boutons nécessaire pour accéder à un ensemble de commandes. Un accès directe à chacune, c'est à dire qui ne nécessite pas l'ouverture d'un menu hiérarchique ou le défilement d'une liste, est toutefois maintenu. En outre, si suffisamment d'espace est disponible, les boutons peuvent être de plus grande taille, ce qui facilite leur sélection.

Rétrocompatibilité

Le canal mixte verre+peau permet facilement de maintenir la compatibilité avec les interactions traditionnelles. L'index est le doigt majoritairement utilisé pour interagir avec un écran tactile. On peut donc choisir de l'affecter aux comportements par défaut. Les nouvelles possibilités introduites par le canal mixte verre+peau ne gênent alors pas les utilisateurs novices qui continuent de manipuler le système comme ils en ont l'habitude. Toutefois, un utilisateur plus expérimenté pourra accéder à un ensemble de commandes supplémentaires (quatre par boutons) à l'aide de ses autres doigts. Comme discuté dans la Partie 5.2, la rétrocompatibilité est susceptible de favoriser l'adoption d'une nouvelle technique d'interaction.

Entrée versus sortie

Si un bouton peut invoquer différentes commandes, il est important de pouvoir communiquer les différentes options qu'il offre. Comme noté précédemment, l'interaction sur tablette tactile est généralement plus limitée en entrée (la surface tactile) qu'en sortie (l'affichage). En effet, la haute résolution des écrans modernes permet de reconnaître aisément de petites icônes bien au-dessous de la taille critique compatible avec une sélection par le doigt. Il est donc possible de montrer l'ensemble des commandes en affichant plusieurs icônes par bouton (une pour chaque commande, voir la Figure 6.4). On peut faire l'analogie avec les claviers physiques qui exposent déjà plusieurs symboles par touches.

6.3.2 Menus

Le canal mixte verre+peau repousse le besoin de recourir à des menus. Toutefois, lorsque le nombre de fonctionnalités devient trop important, il devient difficile de les éviter. Il sont notamment utiles pour organiser les commandes. Cette section examine comment les menus peuvent tirer parti du canal mixte verre+peau.

Raccourcis Les raccourcis, comme les raccourcis clavier (ou *hotkeys*), ne sont généralement pas présents sur les appareils mobiles faute d'un clavier physique. Nous proposons d'utiliser l'identification des doigts comme un substitut pour les raccourcis clavier sur écran tactile. Le canal mixte verre+peau peut permettre à la fois d'interagir de la manière habituelle (ouvrir les menus en cliquant avec l'index sur leur bouton d'ouverture) et d'exécuter des commandes fréquentes plus rapidement en touchant l'écran avec un doigt différent (voir la Figure 6.5).

Dossiers Le même types de raccourcis peut être utilisés sur les dossiers des interfaces mobiles. Les dossiers des interfaces mobiles permettent de regrouper des icônes d'applications. Pour ouvrir une application, il faut dans un premier temps taper sur l'icône du dossier puis taper à nouveau sur l'icône de l'application (une technique d'interaction non sans ressemblance avec *TapTap* [182]). Des raccourcis digitaux peuvent permettre de lancer une application sans nécessiter l'ouverture du dossier. Incidemment, les interfaces Apple représentent un dossier par la miniature des icônes des premières applications qu'il contient, ce qui informe l'utilisateur sur ces différentes alternatives.

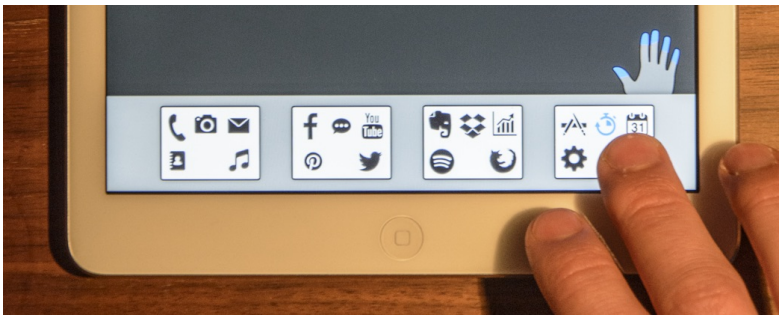
Délimiteur d'activation Les menus contextuels sont souvent ouverts en marquant une pause après avoir touché la surface ce qui permet de différentier leur activation des autres interactions possibles avec l'application. Ceci introduit un délai qui ralentit l'interaction. Il est possible de supprimer ce délai d'ouverture en sacrifiant

un des raccourcis digitaux. Par exemple, le majeur peut servir à ouvrir le menu instantanément.

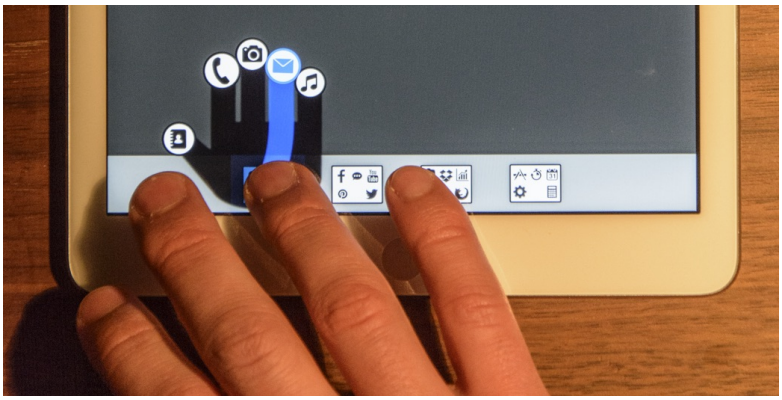
6.3.3 Communiquer l'interaction verre+peau

Découvrabilité

Certains utilisateurs peuvent ne pas être au courant de la disponibilité d'une nouvelle modalité d'interaction reposant sur l'utilisation de différents doigts. Il est donc nécessaire de donner des indices pour les aider à la découvrir sans recourir à un tutoriel vidéo ou à la documentation. La Figure 6.4a montre un exemple de méthode statique. Une main fantôme est affichée au-dessus de la barre d'outils dans le but d'indiquer que différentes actions sont associées à différents doigts. Un exemple de retour dynamique est donné par la Figure 6.4b. Il s'agit d'une courte animation montrant le résultat du contact avec les différents doigts.



(a) Boutons multifonctions de taille importante



(b) Boutons multifonctions de petite taille avec aperçu lors du contact

Figure 6.4 – Exemples de boutons multifonctions

Association doigts-commandes

Il est important de communiquer quel doigt active quelle commande, afin qu'un utilisateur puisse découvrir les différentes actions possibles avec un bouton multi-fonctions. Les figures 6.4 et 6.5 illustrent trois types de retours visuels permettant d'informer sur l'association doigt-commande. Le premier (Figure 6.4a) utilise l'emplacement de l'icône à l'intérieur du bouton afin d'indiquer le doigt cible. Le second (Figure 6.4b) s'inspire de la précédente représentation, mais l'association n'apparaît que sur demande : les utilisateurs doivent presser le bouton et marquer une pause pour l'afficher. Cette seconde approche permet de réduire la taille des boutons ainsi que la quantité d'information affichée à l'écran (et donc le bruit visuel), mais peut s'avérer moins intuitive pour un utilisateur novice. Le dernier exemple (Figure 6.5) utilise les doigts comme raccourcis pour un menu. Un symbole représentant le doigt associé à une commande est affiché à droite de son nom à la manière des raccourcis clavier dans les menus linéaires traditionnels des ordinateurs de bureau. Cette dernière méthode s'inspire des travaux de Appert et Zhai qui intégrèrent à des menus linéaires des pictogrammes représentant des raccourcis gestuels associés aux commandes [7].

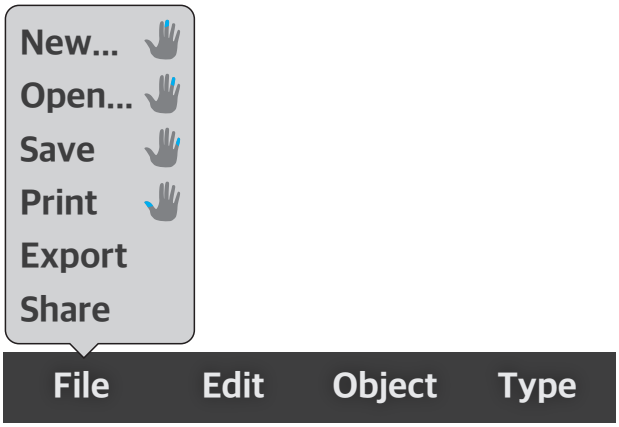


Figure 6.5 – Exemple de menus avec raccourcis digitaux

6.4 Limites

Le canal verre+peau a aussi des limites. En particulier, les différentes techniques d'interaction ne sont pas toutes compatibles les unes avec les autres : par exemple, un bouton verre+peau ne peut pas à la fois lancer cinq applications et ouvrir un menu. Les concepteurs doivent faire des compromis en fonction des besoins des utilisateurs et la cohérence avec les autres applications (ou le système d'exploitation).

Dans certaines situations, il peut être difficile de tirer profit des cinq doigts de la main. Par exemple, certains utilisateurs interagissent et tiennent leur smartphone avec la même main. Il se servent alors du pouce pour pointer. Dans ce cas, non seulement le canal verre+peau est indisponible pour l'utilisateur, mais des erreurs peuvent survenir si l'application ne considère pas le pouce comme le doigt par défaut. Une solution consisterait à affecter les mêmes actions à la fois sur l'index et le pouce. Ceci réduirait toutefois l'expressivité du canal. Une autre solution pourrait être de détecter la manière dont le téléphone est tenu, à la manière de Harrison et al. [93], afin d'éviter les activations accidentelles. On peut noter que ce problème ne se pose pas pour les tablettes tactiles ou les montres pour lesquelles l'interaction au pouce est peu courante De plus, la tendance des téléphones actuels est aux grands écrans. Leur taille ne leur permet donc plus d'être manipulés à une seule main.

6.5 Étude : Pourquoi et quand utiliser le canal verre+peau ?

Nous avons comparé le canal mixte verre+peau et le canal verre (canal tactile traditionnel) par le biais d'une expérience conçue dans l'objectif de mesurer des courbes de débit au sens de Shannon [200]. Le débit (noté TP pour l'anglais *throughput*) est une métrique issue de la théorie de l'information qui permet de mesure la quantité d'information (en bits/s) qui transite à travers un canal de communication. Dans notre cas, la source de ce canal est l'utilisateur et sa destination le système. Plus l'ensemble de commandes est important, plus l'entropie du vocabulaire, c'est-à-dire le potentiel d'information moyen qu'il contient, est grand. Pour plus de simplicité nous supposerons dans ce chapitre que les commandes sont équiprobables. Dans ce cas, l'entropie en entrée (ou l'entropie du vocabulaire H_V) est seulement le log_2 du nombre de commandes possibles. Nous reviendrons plus en détail sur la notion de débit et le calcul de l'information au sens de Shannon dans la Section 6.5.1.

Nous avons formulé trois hypothèses :

1. Quand l'entropie du vocabulaire augmente, la quantité d'information transmise doit se stabiliser et atteindre un plateau comme c'est le cas dans des tâches de jugement absolu [152].
2. D'autre part, le temps de sélection moyen μ_T doit augmenter à peu près linéairement avec H_V, en raison de la loi de Hick [98] et la loi de Fitts [70].
3. Il résulte de ces deux premières hypothèses que la dépendance du débit TP à l'entropie du vocabulaire H_V doit être en forme de cloche. En conséquence, pour tout canal d'entrée donné, il doit exister un niveau d'entropie optimale.

Ainsi nous allons nous concentrer sur le débit maximum (TP_{max}). On désigne

par niveau d'entropie optimal H_{opt} le niveau particulier de l'entropie où TP_{max} est atteint. On obtient deux métriques empiriques distinctes :

— plus grand est TP_{max}, plus la quantité d'information transmise en un temps donné est importante,

— plus grand est H_{opt}, plus grande est la taille du vocabulaire utilisable.

Avec le canal verre+peau, les utilisateurs contrôlent non seulement la sélection d'une région de l'écran, mais également la sélection d'une région de la surface de leur propre corps. La surface du verre et la surface de la peau devraient donc être plus ou moins utilisables comme deux canaux d'entrée indépendants. En conséquence, à la fois TP_{max} et H_{opt} devraient être plus élevés pour le canal mixte verre+peau que pour notre référence, le canal verre.

6.5.1 L'Interaction utilisateur vue comme débit d'information

Lors d'une expérience utilisateur, les performances sont traditionnellement analysées sur deux axes :

1. En termes de vitesse ou de temps moyen nécessaire pour effectuer une tâche.
2. En termes de taux d'erreurs ou de précision.

Or ces deux variables sont fortement corrélées : plus un utilisateur réalise une tâche rapidement, plus son taux d'erreurs augmente, et inversement [92, 150, 167, 203]. Il s'agit d'un résultat classique de psychologie expérimentale. Ainsi, il est risqué d'affirmer qu'une technique T_1 est plus rapide qu'une technique T_2 au prix d'un taux d'erreurs plus élevé. En situation réelle, un utilisateur exposé à un fort taux d'erreurs adaptera souvent sa vitesse d'exécution afin d'améliorer son taux de succès. Dans ce cas, le couple temps/erreur ne permet pas de conclure sur les performances relatives des techniques T_1 et T_2.

La théorie de l'information, dont les bases ont été édifiées par Claude Shannon en 1948 [200], permet de quantifier le contenu en information d'un ensemble de messages, ainsi que l'information transmise par un canal de communication en tenant compte du bruit (c'est-à-dire les perturbations venant altérer la transmission des messages).

L'Interaction Homme-Machine peut être vue comme un transfert d'information entre l'homme et le système. Si l'on est capable de modéliser le *canal interactionnel* et en particulier sa source et sa destination, nous sommes en mesure de calculer son débit, en bits/s, au sens de Shannon. Le débit rassemble en une seule et même métrique le temps d'exécution et le taux d'erreurs. Il dépend du nombre de commandes pouvant être exécutées ce qui nous intéresse particulièrement pour la présente expérience. Nous proposons donc de l'utiliser comme nouvelle métrique expérimentale en IHM afin d'estimer les performances d'une technique d'interaction ou d'un dispositif d'entrée. Nous avons suivi cette approche lors de notre expérience.

L'objectif de cette partie est de montrer comment l'information au sens de Shannon peut être exploitée comme métrique en IHM et de poser les bases nécessaires à l'analyse des résultats de notre expérience. Elle explicitera brièvement les notions de théories de l'information nécessaires en prenant pour référence principale « *Théorie de l'information et du codage* » de Oliver Rioul [177], un des acteurs impliqués dans la présente étude.

Le canal interactionnel

La première étape pour mesurer la bande passante du canal de l'interaction entre un utilisateur et un système est de définir précisément quelle est l'entrée et la sortie de ce canal. Dans ce manuscrit, nous nous intéressons essentiellement à l'Interaction Humain-Machine en entrée, c'est-à-dire à la sélection de commandes ou à la manipulation de paramètres par exemple.

Par définition, pour l'interaction en entrée, la source du canal de l'interaction humain-machine est l'utilisateur (l'humain) et sa sortie le système auquel il donne des ordres (la machine). Nous définissons les messages en entrée de ce canal comme les intentions de l'utilisateur, par exemple, « ajouter un carré », « se déplacer vers le haut de l'image », « diminuer le contraste », etc. Nous définissons les messages en sortie comme la réaction du système, notamment sur les données où les objets d'intérêts. Le canal de l'interaction humain-machine en entrée est représenté sur la Figure 6.6. À l'idéal, l'entrée et la sortie du canal forme une bijection : chaque intention utilisateur résulte systématiquement en la commande système appropriée. En d'autres termes la totalité de l'information portée par les intentions de l'utilisateur est transmise au système. Toutefois, ce canal est bruité, notamment par des erreurs système ou utilisateur. En conséquence, les actions systèmes ne correspondent pas toujours à l'intention originale de l'utilisateur et seulement une partie de l'information est transmise.

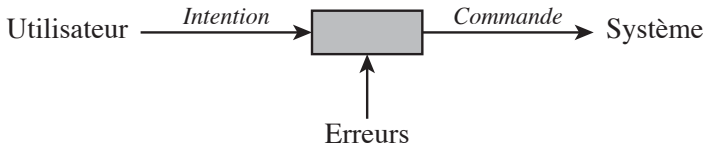


Figure 6.6 – Le canal de l'interaction humain-machine en entrée

Information selon Shannon

Selon Shannon [200], donner de l'information revient à lever de l'incertitude. Calculer l'information d'un message (aussi appelé symbole) correspond donc à un calcul de probabilité : plus une information est incertaine, plus elle est intéressante, et une information certaine, que nous connaissions donc déjà, n'a aucune valeur.

Formulé autrement, plus on s'attend à voir apparaître un message, moins il nous apporte d'information, et au contraire, plus un message est inattendu, plus il est informatif. L'information I portée par un message x est définie par la formule suivante :

$$I(x) = log_2 \frac{1}{p(x)} \tag{6.1}$$

Aux extrêmes, pour $p(x) = 1$ on retrouve $I(x) = 0$ et pour $p(x) = 0$, on a $\lim_{p(x)\to 0} I(x) = \lim_{p(x)\to 0} \log_2 \frac{1}{p(x)} = +\infty$ (il est infiniment surprenant de voir arriver un message qui n'apparait jamais). En IHM, ces messages correspondent à l'intention de l'utilisateur en entrée, et à l'exécution d'une commande en sortie (voir le schéma 6.6).

La production de ces messages peut être assimilée à une variable aléatoire notée X. Il s'agit de l'entrée de notre canal interactionnel. L'ensemble des messages possibles est appelé *alphabet* et est noté $\mathcal{X}$. En IHM, cet alphabet correspond au vocabulaire de commandes.

Entropie L'information moyenne de X est appelées *entropie* et est notée H :

$$H(X) = \mathbf{E}\, I(X) = \sum_x p(x) \log_2 \frac{1}{p(x)} \tag{6.2}$$

Cela correspond donc à l'information moyenne des possibilités offertes à l'utilisateur.

Dans le cas particulier où chaque message est équiprobable, on a $p(x) = \frac{1}{N}$ et donc $H(X) = \log N$ où N correspond à la taille de l'alphabet $\mathcal{X}$. Il s'agit d'un cas que l'on retrouve régulièrement lors des expériences utilisateurs.

Information transmise au sein d'un canal L'entrée et la sortie d'un canal C sont considérées comme des variables aléatoires. L'information transmise au sein de ce canal correspond à l'information mutuelle de l'entrée X et de la sortie Y. Elle est mesurée en bits et est calculée comme suit :

$$I_t(C) = I(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{6.3}$$

On peut également démontrer les égalités suivantes permettant de lier l'information transmise à l'entropie de X et de Y [177] :

$$I_t(C) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{6.4}$$

Si l'information est transmise parfaitement, la connaissance de Y n'apporte aucune information si l'on connait déjà X, et vice-versa. On a alors $H(X|Y) = H(Y|X) = 0$ (cela peut se prouver mathématiquement [177]) et donc $I(X,Y) = H(X) = H(Y)$. En IHM cela correspond au cas où il n'y a jamais aucune erreur.

Débit Le débit TP (pour l'anglais *throughput*), mesurée en bit/secondes correspond à l'information moyenne transmise par un canal I_t, divisée par le temps moyen T entre chaque message :

$$TP = \frac{I_t}{T} \tag{6.5}$$

Mise en œuvre

Lors d'une expérience contrôlée, les stimulus permettent de simuler l'intention de l'utilisateur. La source X du canal est donc donnée par l'apparition de ces stimulus. Les probabilités p sont estimées à partir de la fréquence d'apparition des stimulus et des commandes et il est donc possible de calculer l'information transmise à l'aide de la formule 6.3.

Il est important de noter que l'information transmise est calculée à partir des stimulus qui ont été *présentés* au participant durant l'expérience. En conséquence, le nombre de fois où chacun apparaît doit être maîtrisé afin de correspondre exactement aux fréquences réelles ou recherchées. Il s'agit d'une contrainte forte, ce paramètre a un impact important sur l'information transmise qui sera calculée par la suite. Il n'est pas possible de tester un participant sur une partie seulement des stimulus par exemple. Pour mesurer le débit maximal possible avec un canal, on choisira un jeu de stimulus équiprobables (il s'agit du cas maximisant l'information en entrée). On présentera alors à chaque participant chaque stimulus exactement le même nombre de fois.

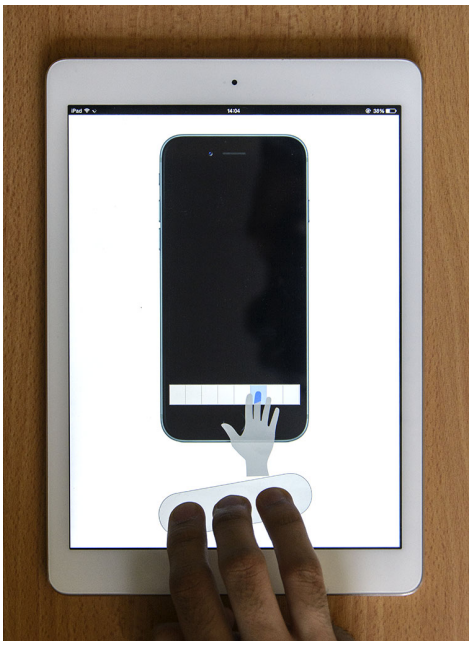
L'entrée et la sortie du canal interactionnel ne sont pas toujours les mêmes d'une expérience à une autre en fonction du canal qui intéresse l'expérimentateur. Par exemple, une tâche de Fitts s'intéresse en général uniquement au temps moteur et supprime le temps de réaction. Le canal mesuré correspond au système moteur et son entrée se situe donc en aval de l'intention utilisateur.

6.5.2 Participants et dispositif

14 droitiers (dont 5 femmes) âgés de 21 à 33 ans acceptèrent de participer à cette expérience. Ils furent recrutés parmi les étudiants de notre laboratoire et de notre école.

L'équipement utilisé consistait en une tablette iPad (24,6 cm de diagonale) sur laquelle était simulé un iPhone (voir Figure 6.7). Un bouton de démarrage était affiché sous la représentation du téléphone. Les participants devaient le presser à l'aide de l'index, du majeur et de l'annulaire de leur main droite, ce qui permettait de standardiser à la fois la position et la posture. La zone d'interaction cible s'étendait sur toute la largeur du téléphone (59 mm) et mesurait 0,9 mm de hauteur, ce qui correspond à la taille des barres d'outils ou *docks* traditionnels sur les *smartphones* (notamment ceux d'Apple qui ont servi de patron). Dans des études pilotes nous avons fait varier la taille de la zone cible et la disposition des boutons, mais nous

(a) Stimulus lors de la condition canal tactile traditionnel



(b) Stimulus lors de la condition canal mixte verre+peau

Figure 6.7 – Affichage et apparence des stimulus

avons décidé de nous concentrer sur une configuration plus simple afin de garder l'expérience suffisamment courte. Ces études pilotes, dans lesquelles nous avons également exploré des dispositions de boutons en deux dimensions, ont montré qu'une organisation 1D produit essentiellement le même résultat. Le logiciel a été développé en JavaScript.

6.5.3 Méthode

Tâche et stimulus

Après l'affichage du stimulus, il était demandé aux participants de presser un bouton aussi vite et aussi précisément que possible. Dans la condition canal tactile traditionnel, le stimulus consistait à un surlignage bleu du bouton cible (Figure 6.7a). Dans la condition canal mixte verre+peau, une main montrant le doigt à utiliser était affichée en complément (Figure 6.7b). Le doigt à utiliser était également indiqué en bleu et la main était positionnée de telle sorte que le bon doigt soit positionné au-dessus du bouton cible.

Procédure

Les participants démarraient un essai en plaçant leurs trois doigts longs (index, majeur et annulaire) sur le bouton de départ sous la représentation du téléphone (Figure 6.7). Un stimulus dépendant de la condition était alors présenté. Ce stimulus restait affiché jusqu'à ce que le bouton de démarrage soit relâché.

Si le bouton cible était correctement sélectionné, il passait au vert. Si un mauvais bouton était atteint, ce dernier passait au rouge. Si aucun bouton n'était pressé, le participant devait presser à nouveau le bouton de démarrage afin de redémarrer l'essai.

En pratique, l'identité des doigts n'était pas capturée par notre système. Des enregistrements vidéos provenant de pilotes où le même type de stimulus avait été utilisé montrèrent un taux d'erreurs particulièrement faible concernant l'utilisation des doigts (2,3 % en moyenne, $\sigma = 2{,}0\,\%$). Pendant l'expérience, un échantillon de 3 participants fut également filmé et révéla un résultat similaire (1,5 % en moyenne, $\sigma = 0{,}52\,\%$).

Les différents facteurs furent testés dans un plan d'expérience intra-participant. L'ordre des canaux et l'ordre des différentes tailles de vocabulaire de commandes furent contre balancés entre participants à l'aide d'un carré latin. Chaque commande apparaissait aléatoirement 3 fois par bloc. L'expérience durait environ 30 min par participant. Nous avons en tout enregistré $14\,\text{participants} \times 3\,\text{iterations} \times (5 + 10 + 15 + 20 + 30 + 40 + 5 + 10 + 20 + 30 + 40 + 50 + 70) = 14\ 490\,\text{essais}$.

Taille du vocabulaire

En nous basant sur les données de nos pilotes, nous avons choisi d'utiliser des vocabulaires de 5, 10, 15, 20, 30, et 40 commandes différentes pour la condition canal verre et de 5, 10, 20, 30, 40, 50 et 70 commandes différentes pour la condition canal mixte verre+peau. Nous avons choisi une surface d'interaction fixe de $59 \times 0{,}9\,\text{mm}$ ce qui correspond à la taille traditionnelle d'une barre d'outils sur un iPhone. Logiquement, plus le nombre de commandes augmente, plus la largeur des boutons s'amenuise. Intrinsèquement, dans la condition canal verre+peau, 5 fois moins de boutons étaient nécessaires pour atteindre la même taille de vocabulaire (voir le tableau 6.1).

6.5.4 Analyse des résultats

Les résultats de cette expérience ont fait l'objet de deux analyses distinctes. La première reprend la méthodologie classique de la recherche en IHM et s'intéresse au temps de réaction, au temps de mouvement, au temps de sélection et au taux d'erreurs. Lors de la seconde, nous avons analysé nos données à la lumière de la théorie de l'information au sens de Shannon [200] et nous nous sommes intéressés à mesurer le débit des deux différents canaux.

Tableau 6.1 – Nombre de boutons, et largeur des boutons en fonction du type d'interaction et du nombre de commandes disponibles

	Canal verre		Canal mixte verre+peau	
Nombre de commandes	Nombre de boutons	Largeur des boutons	Nombre de boutons	Largeur des boutons
5	5	12 mm	1	58 mm
10	10	5,8 mm	2	29 mm
15	15	3,9 mm	-	-
20	20	2,9 mm	4	15 mm
30	30	1,9 mm	6	9,7 mm
40	40	1,4 mm	8	7,3 mm
50	-	-	10	5,8 mm
70	-	-	14	4,2 mm

Analyse classique temps et erreur

Les variables dépendantes qui nous intéressent sont :

— le temps de réaction (TR correspondant au temps écoulé entre l'apparition du stimulus et le moment où le bouton de démarrage est relâché),

— le temps de mouvement (TM correspondant au temps écoulé entre le relâchement du bouton de démarrage et le premier contact avec la surface tactile),

— le temps de sélection total ($TT = TR + TM$),

— le taux d'erreurs.

Les valeurs non communes du facteur nombre de commandes (c'est-à-dire 15, 50 et 70, voir le tableau 6.1) furent ignorées pour l'analyse de temps et d'erreur afin que la comparaison entre le canal verre et le canal mixte verre+peau soit pertinente. La signification statistique des résultats fut évaluée en utilisant des ANOVAs.

Temps de réaction. Le temps de réaction (TR) fut plus court avec le canal verre qu'avec le canal mixte verre+peau (Figure 6.8), un résultat observé sur l'ensemble des 14 participants ($F_{1,13} = 76$, $p < 10^{-3}$). La différence moyenne, calculée sur les abscisses communes, fut de 132 ms. Le nombre de commandes affecta légèrement TR pour le canal verre, mais pas pour le canal mixte verre+peau.

Temps de mouvement. Dans l'ensemble, le temps de mouvement fut plus court avec le canal mixte verre+peau qu'avec le canal verre (Figure 6.9). La différence moyenne fut de 43 ms. L'effet de la taille du vocabulaire, plus prononcé pour TM

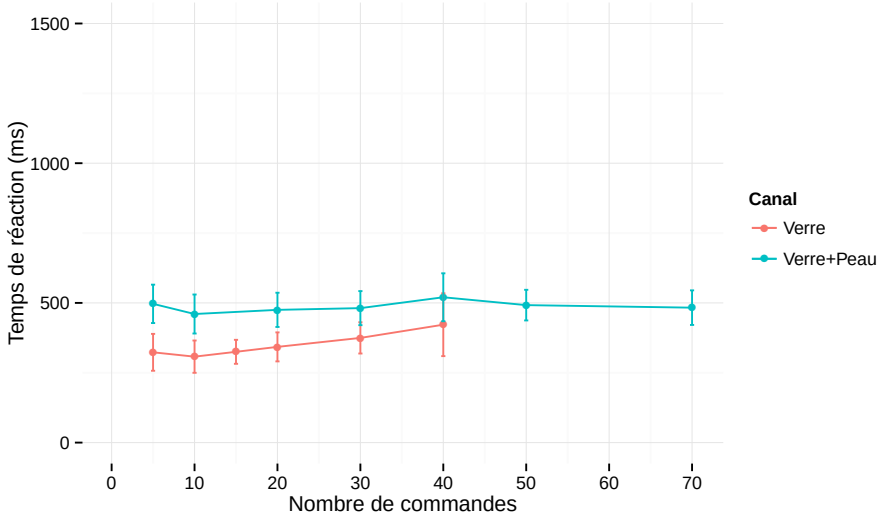


Figure 6.8 – Temps de réaction (les valeurs en abscisse non communes aux deux canaux furent ignorées lors de l'analyse statistique)

que pour TR, fut approximativement linéaire avec une pente plus raide pour le canal verre ($F_{1,13} = 26$, $p < 10^{-3}$).

Temps de sélection total. En moyenne, sur l'intervalle d'abscisses communes investigué pendant cette expérience, le temps de sélection total ($TT = TR + TM$) fut légèrement plus long (89 ms) avec le canal mixte verre+peau (Figure 6.10). Toutefois, plus important, ce temps s'accroît également beaucoup plus lentement avec la taille du vocabulaire avec le canal mixte verre+peau ($F_{1,13} = 16$, $p < 10^{-2}$). À partir de 30 commandes, ce dernier devient plus rapide.

taux d'erreurs. Enfin, le taux d'erreurs augmente considérablement plus lentement avec le canal mixte verre+peau ce qui indique la possibilité de manipuler des ensembles de commandes bien plus importants (Figure 6.11). Bien que ce taux d'erreurs n'inclut pas les erreurs de doigts, nos enregistrements vidéos (pris sur un échantillon de 3 participants) ont montré que celles-ci étaient rares (1,5 % des essais en moyenne, $\sigma = 0{,}52\,\%$).

Pour conclure cette analyse classique de nos données, prendre en compte l'identité des doigts permet d'améliorer à la fois la vitesse et la précision avec un vocabulaire de commandes de taille importante.

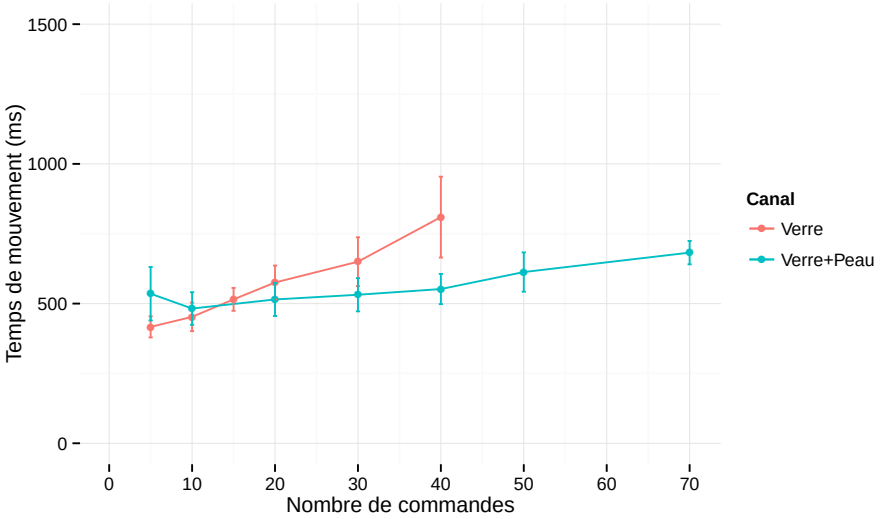


Figure 6.9 – Temps de mouvement (les valeurs en abscisse non communes aux deux canaux furent ignorées lors de l'analyse statistique)

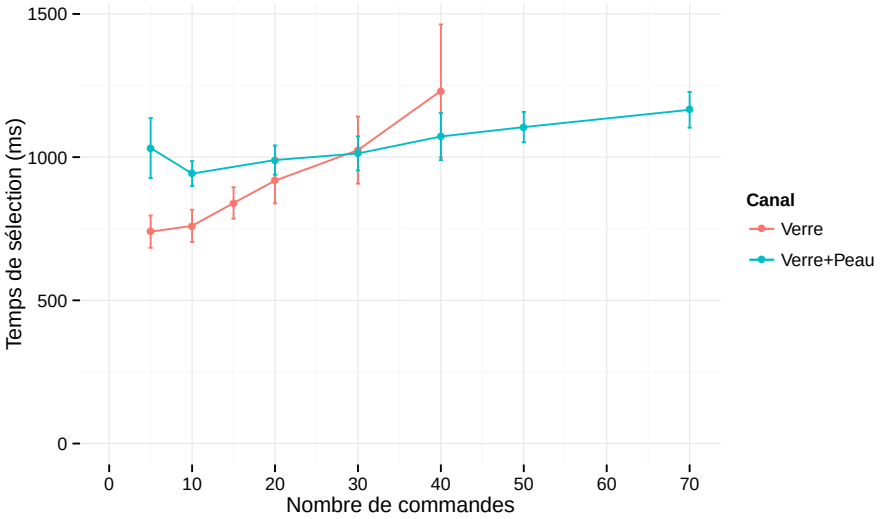


Figure 6.10 – Temps de sélection total (les valeurs en abscisse non communes aux deux canaux furent ignorées lors de l'analyse statistique)

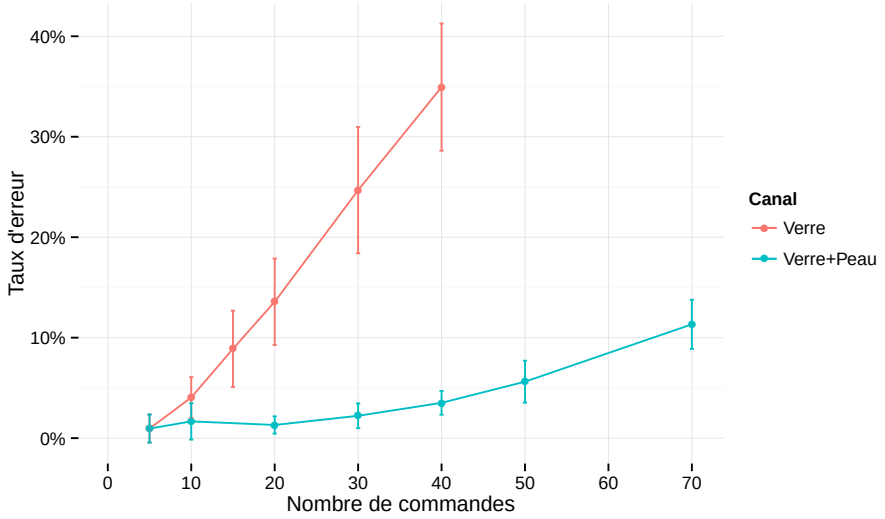


Figure 6.11 – taux d'erreurs (les valeurs en abscisse non communes aux deux canaux furent ignorées lors de l'analyse statistique)

Analyse du point de vue de la théorie de l'information

La raison principale qui nous a amené à nous intéresser une analyse du débit (TP) est que cette mesure permet de regrouper à la fois la vitesse et la précision en une seule quantité. Nous nous sommes donc demandé comment la quantité d'information transmise I_t (en bits), puis le débit TP (en bit/s), variaient en fonction de l'entropie du vocabulaire H_V. Les stimulus en entrée étant équiprobables, l'entropie du vocabulaire $H_V = log_2(N)$ où N représente la taille du vocabulaire. Le tableau 6.2 donne l'entropie des différents vocabulaires utilisés en fonction du nombre de commandes.

Information Transmise. Pour les deux conditions, l'information transmise I_t a tendance à se stabiliser alors que l'entropie H_V s'élève (Figure 6.12). Ce constat tend à confirmer que les deux canaux testés ont une capacité (en bits par sélection) limitée. Cette stabilisation aurait sans doute été encore plus marquée s'il nous avait été possible d'explorer des vocabulaires plus larges. Toutefois, l'étude d'un vocabulaire avec un haut niveau entropique prend d'une part beaucoup de temps, mais est de plus particulièrement frustrante pour les participants à cause de la difficulté considérable de la tâche et le fort taux d'erreurs qui en découle. Comme nous le verrons pas la suite, cet intervalle d'abscisse, choisi suite aux résultats de nos pilotes, nous permet bien d'observer adéquatement les courbes de débit.

Les deux courbes représentées sur la Figure 6.12 suggèrent toutes deux une tendance asymptotique en deux différentes capacités limites. Avec le canal mixte

Tableau 6.2 – Entropie en fonction du nombre de commandes

Nombre de commandes	Entropie du vocabulaire
5	$\sim$ 2,322 bits
10	$\sim$ 3,322 bits
15	$\sim$ 3,907 bits
20	$\sim$ 4,322 bits
30	$\sim$ 4,907 bits
40	$\sim$ 5,322 bits
50	$\sim$ 5,644 bits
70	$\sim$ 6,129 bits

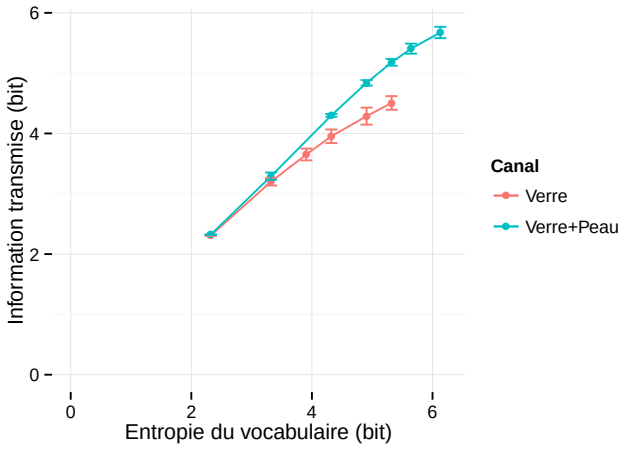


Figure 6.12 – Information transmise

verre+peau, non seulement la quantité moyenne d'information transmise était plus importante qu'avec le canal tactile traditionnel (une différence observée sur l'ensemble des 14 participants), mais également, la position de l'asymptote suggère que le sommet de la courbe est plus haut.

Débit. Comme prédit, le débit TP atteint un maximum avec les deux conditions (Figure 6.13). À l'aide d'une régression polynomiale du second ordre sur nos données, nous avons pu obtenir les équations suivantes :

$$y = -0{,}389x^2 + 3{,}2043x - 2{,}1714 \tag{6.6}$$

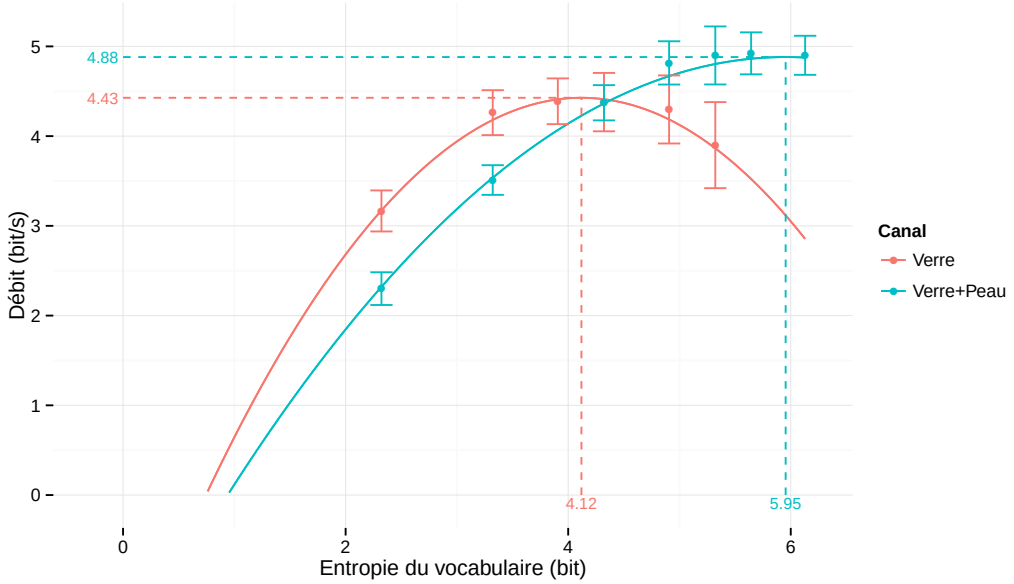


Figure 6.13 – Courbes de débit

pour le canal tactile ($r^2 = 0{,}985$) et

$$y = -0{,}1941x^2 + 2{,}3115x - 2{,}0004 \tag{6.7}$$

pour le canal mixte verre+peau ($r^2 = 0{,}997$). À partir de ces deux équations, représentées sur la Figure 6.13, nous avons pu estimer les coordonnées XY des débits maximums possibles (les deux maximums sont situés sur l'intervalle d'abscisse testé et donc aucune extrapolation ne fut nécessaire) :

— pour le canal tactile, $TP_{max} = 4{,}43$ bits/ sec atteint pour un niveau d'entropie de 4,12 bits et

— pour le canal mixte verre+peau, $TP_{max} = 4{,}88$ bits/ sec atteint pour un niveau d'entropie de 5,95 bits.

Ainsi, le débit permet de montrer sans ambiguïté que le canal mixte verre+peau apporte deux améliorations indépendantes. Tout d'abord, il apporte un accroissement du débit de 10,1 %, ce qui montre une transmission de l'information plus efficace de l'utilisateur au système. En second lieu, le débit maximal est atteint à un niveau d'entropie plus élevé : l'optimum d'entropie s'est donc déplacé vers la droite ce qui signifie que des ensembles de commandes substantiellement plus importants peuvent être manipulés efficacement.

Différences de performance entre les doigts

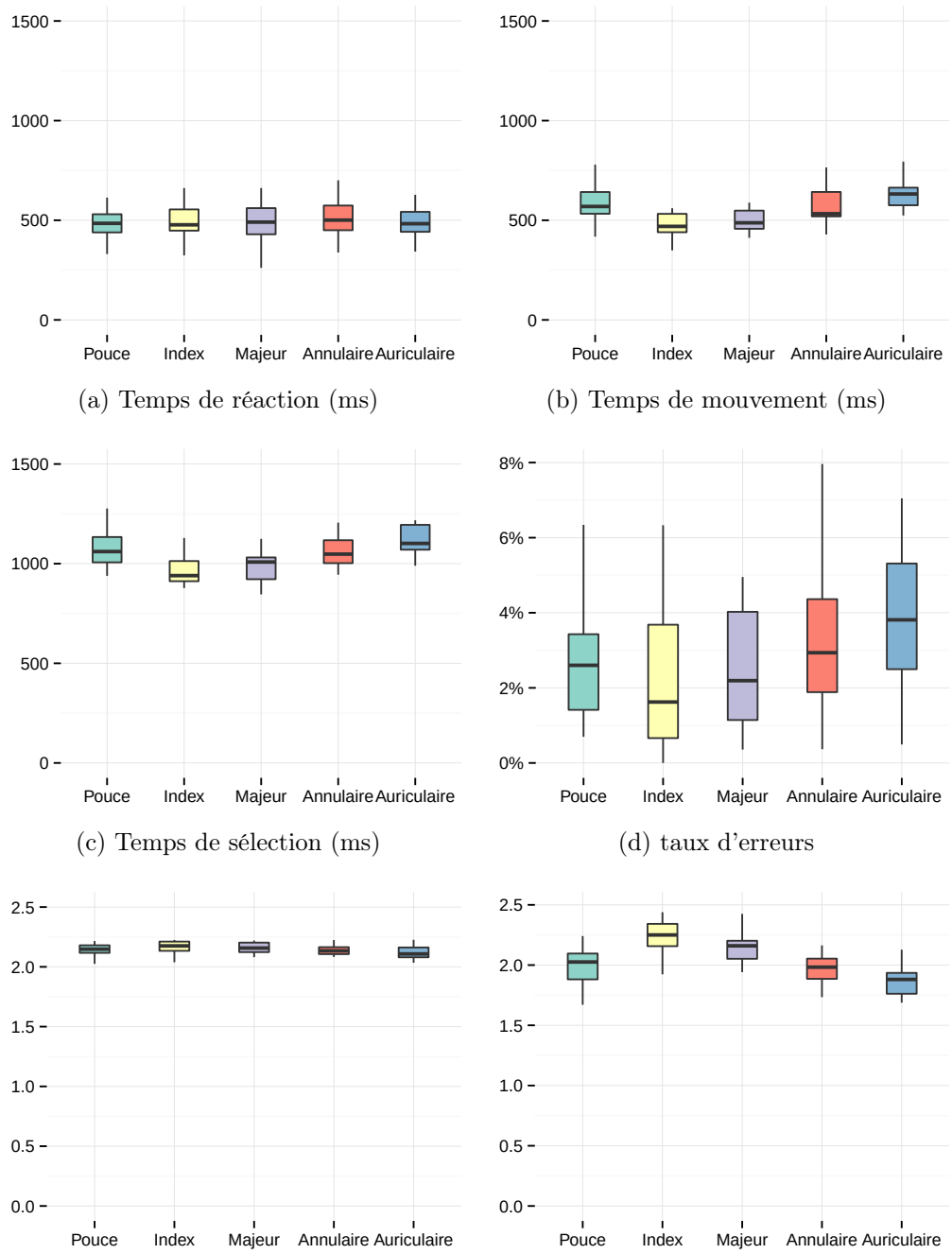
Les performances des doigts ne sont pas égales. La Figure 6.14 montre, sans surprise, que le doigt le plus performant est l'index et que le moins performant est l'auriculaire. Cette différence se constate particulièrement sur les données de débit de la Figure 6.14f. Ces données peuvent être prises en compte lors de la conception de techniques d'interaction exploitant le canal mixte verre+peau, en particulier en tirant davantage parti des trois doigts longs. On peut noter que même en supprimant le pouce et l'auriculaire, la possibilité de multiplier le vocabulaire par 3 (soit ajouter $log_2(3) = 1{,}58$ bits) semble déjà d'un intérêt non négligeable.

6.6 Conclusion du chapitre

En considérant la littérature actuelle, peu de doute demeure sur le potentiel du canal mixte verre+peau pour améliorer significativement l'expressivité des écrans tactiles. Notre expérience semble indiquer qu'assez peu d'effort est nécessaire pour atteindre une cible avec un doigt particulier. Nos pilotes semblent également montrer qu'il est rare que l'utilisateur se trompe de doigt.

L'exploitation du canal mixte verre+peau parallèlement au canal tactile habituel permet de limiter la consommation d'espace à l'écran, une ressource précieuse sur les dispositifs mobiles. Ceci peut aussi permettre d'augmenter la largeur de la hiérarchie des commandes et donc de limiter sa profondeur, dont l'exploration est coûteuse en temps. L'apport du canal mixte verre+peau est loin de se limiter aux tablettes. Par exemple, même avec seulement trois doigts, il pourrait apporter une solution efficace et élégante pour interagir avec la surface interactive réduite des *smartwatches*.

Lors de cette expérience, nous avons offert une vue schématique théorique des possibilités de ce nouveau canal. Nous avons volontairement ignoré le fait qu'en situation réelle, certaines commandes sont bien plus fréquentes que d'autres, ce qui veut dire que le niveau d'entropie en entrée est en réalité bien inférieur à celui que nous avons simulé. En général, on réservera les commandes plus fréquentes aux doigts avec le meilleur débit (l'index et le majeur comme l'indique la Figure 6.14) et on laissera aux doigts les plus lents les commandes les plus rares (voire éventuellement aucune). Ces questions sont subtiles mais importantes et méritent d'être traitées plus en profondeur lors de futurs travaux.



(a) Temps de réaction (ms)

(b) Temps de mouvement (ms)

(c) Temps de sélection (ms)

(d) taux d'erreurs

(e) Information transmise (bit)

(f) Débit (bit/sec)

Figure 6.14 – Performances entre doigts

Conclusion

Notre objectif général dans ce travail a été de contribuer à enrichir l'interaction sur tablettes tactiles en rendant utilisables sur ces surfaces réduites un plus grand nombre de fonctionnalités. Le contexte spécifique auquel nous avons songé en priorité est la revue radiologique numérique, une tâche difficile nécessitant l'utilisation et la manipulation d'outils complexes avec une grande précision.

Contributions

À partir d'une étude approfondie des potentialités et des faiblesses des tablettes tactiles, nous avons exploré deux directions qui nous ont semblé particulièrement prometteuses :

1. Mieux exploiter les capacités existantes en entrée en concevant des techniques d'interaction plus expressives. Nous nous sommes en particulier intéressés au problème de la transition entre modes et à l'amélioration des performances des utilisateurs experts.
2. Étendre les capacités de la tablette en exploitant de nouveaux canaux d'entrée.

Tout d'abord, nous avons cherché à clarifier les notions de mode et de délimiteur, en soulignant à quel point les modes sont omniprésents et inévitables dans les interfaces actuelles. La clé pour en faciliter la manipulation est la conception de délimiteurs efficaces, tant pour l'activation d'une technique que la séparation des différentes étapes qui la composent. Nous avons ensuite exploré différentes techniques de transition entre modes en nous appuyant sur notre modèle et en nous inspirant de notre classification des délimiteurs :

1. Le *FingerMenu* qui repose sur un délimiteur d'activation *multitouch* et qui associe à chaque doigt un outil,
2. Le *SigmaMenu* qui repose sur un délimiteur d'activation gestuel circulaire et qui s'inspire du *MarkingMenu*.

Un défaut majeur des techniques standards est de confisquer une proportion substantielle de l'espace écran alors que celui-ci est nécessairement de taille réduite sur une tablette. Les deux techniques proposées résolvent ce problème sans pour autant entraîner de baisse de performance par rapport à des techniques usuelles comme les barres d'outils et les barres de menus.

Nous avons également présenté deux autres techniques d'interaction qui se concentrent plus spécifiquement sur l'utilisation experte :

1. Les Lettres Augmentées qui s'appuient sur le langage pour permettre l'accès à un grand nombre de commandes facilement mémorisables. Pour ce faire, elles associent une lettre, tracée sans relever le doigt, avec un trait directionnel inspiré du *Marking Menu*. Cette lettre est typiquement l'initiale de la commande tandis que le trait directionnel permet de résoudre les conflits de noms. Notre expérience a montré que cette technique permettait de mémoriser environ 25% de commandes supplémentaires sans pour autant impacter la vitesse d'exécution. Cette technique a été publiée et présenté à la conférence CHI en 2013 [185].
2. La Barre de Menus *MultiTouch* est une nouvelle barre de menus, compatible avec les barres de menu traditionnelles, mais qui s'inspire de *FingerCount* [21] pour permettre une sélection plus rapide en tenant compte du nombre de doigts. Notre hypothèse est que sa compatibilité avec les techniques auxquelles sont habitués les utilisateurs favorise son adoption et lui permet d'être facilement intégrée à une application sans nécessiter une refonte du modèle interactionnel de cette application.

Validé en laboratoire, l'ensemble de ces techniques – à l'exception, pour l'instant, des Lettres Augmentées – a été intégré à notre prototype de radiologie.

Pour finir, nous avons présenté une étude à caractère plus fondamental consacrée à la l'opportunité d'exploitater le canal d'entrée verre+peau, lequel devient disponible dès lors que la technologie permet de distinguer le doigt utilisé en plus des coordonnées du canal de pointage usuel. Nous avons montré théoriquement et confirmé expérimentalement que l'exploitation du canal verre+peau permet une nette amélioration des performances pour un nombre de commandes important. Dans cette optique, nous nous sommes appuyés sur une nouvelle métrique expérimentale en IHM : le débit de Shannon [200]. Cette étude a été publiée et présentée à la conférence INTERACT en 2015 [184].

Perspectives

Amélioration et études des techniques proposées

Tout d'abord, nous poursuivons nos travaux sur le *SigmaMenu* et le *FingerMenu*.

Concernant le premier, plusieurs améliorations sont en cours de développement. Notamment, nous voulons remplacer les délimiteurs actuels de la technique, basés sur des distances seuils, par des délimiteurs reposant sur la courbure du geste. Notre hypothèse est que ces nouveaux délimiteurs permettront de réduire la durée de l'enregistrement de la technique (en particulier lors de l'invalidation de son délimiteur d'activation), et d'améliorer son taux d'erreurs.

Nous souhaitons effectuer cette dernière avec un plus grand nombre de commandes dans le double objectif de mieux évaluer les capacités de ce menu ainsi que l'impact

de la présence d'un double sens de rotation. Il serait en outre intéressant d'étudier la capacité des utilisateurs à mémoriser les associations entre les commandes et les gestes requis pour les exécuter.

Nous souhaitons également étudier le facteur mémorisation.

Le *FingerMenu* s'est également avéré particulièrement prometteur. Différentes variantes ont été proposées en Partie 4.1.4 afin d'accroitre la taille des vocabulaires de commandes, par exemple en y adjoignant des gestes directionnels similaires au *Marking Menu*.

Des expériences complémentaires sont nécessaires pour évaluer les capacités de ces extensions ansi que pour valider la technique de barre d'outils *multitouch* présenté en Partie 5.2. Dans ce cadre, nous nous intéressons en particulier à l'apprentissage et la transition de novice à expert.

Étude de terrain et autres cas d'application de la tablette dans le contexte de la radiologie

Durant la thèse, il n'a pas été possible de mettre en place une étude de terrain avec des radiologues. Ceux-ci sont en effet peu disponibles et il est difficile de leur demander de tester un prototype. *General Electric* travaille toutefois à l'élaboration d'un produit commercial pour tablette. À plus long terme, nous aimerions également étudier nos différentes techniques dans un contexte réel, si possible en nous appuyant sur ce produit.

Un cas d'utilisation intéressant à examiner dans la suite de cette thèse est l'utilisation de la tablette comme un périphérique satellite à la station principale. Ce satellite peut permettre d'afficher une vue supplémentaire, de donner un accès rapide à des commandes fréquentes ou d'effectuer certaines opérations plus adaptées à une surface tactile qu'à une souris. Cet axe semble particulièrement prometteur et pourrait être le sujet de futurs travaux.

Études fondamentales et méthodologie

D'un point de vue plus fondamental, nous voudrions continuer notre exploration des délimiteurs. Notre intuition est que les délimiteurs sont des éléments atomiques fondamentaux qui rythment l'interaction avec un système. Lors de cette thèse, nous les avons essentiellement abordés dans le cadre des changements de modes. Toutefois, leur usage ne se limite pas à cela : même l'exécution d'une commande est déclenchée par un délimiteur, par exemple via un clic de la souris ou d'une pression sur un bouton physique. Il est difficile de dégager un espace de conception des délimiteurs en raison de leur très grande diversité. En effet, comme le font remarquer Hinckley et al., il s'agit juste de « quelque chose de différent dans le flux d'entrée » [101]. Chaque nouveau dispositif d'entrée amène de nouveaux éléments pouvant faire office de délimiteur. Cependant, une caractérisation plus approfondie des différents

délimiteurs possibles et de leurs forces et faiblesses (notamment en fonction du contexte d'utilisation et des caractéristiques des modes correspondants) pourrait être d'un grand bénéfice lors de la conception de nouvelles techniques d'interaction De plus, un espace génératif serait bien entendu d'autant plus puissant.

La nouvelle mesure du débit au sens de Shannon [200] que nous avons commencé à manipuler dans le contexte de l'interaction humain-machine semble également avoir un potentiel intéressant pour la discipline. Nous avons vu comment elle pouvait permettre de résoudre les conflits entre le couple erreur/temps d'exécution. En tenant compte de l'information portée par un vocabulaire et donc de sa taille, elle peut également ouvrir la voie à des comparaisons de techniques à tailles de vocabulaire inégales. Aujourd'hui, il est difficile de comparer les performances d'une barre de menus et d'une barre d'outils par exemple : la première est inévitablement plus lente que la seconde, mais permet l'accès à bien davantage de commandes. Le débit est une métrique qui peut nous permettre de les comparer en tenant compte de ces deux facteurs. La manière d'interpréter ce résultat et ses implications concrètes restent toutefois à l'étude.

Bibliographie

[1] Johnny Accot et Shumin Zhai. « More than dotting the i's — foundations for crossing-based interfaces ». In : *CHI '02*. New York, New York, USA : ACM, avr. 2002, p. 73–80 (cf. p. 59, 60).

[2] Anand Agarawala et Ravin Balakrishnan. « Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen ». In : *CHI '06*. New York, NY, USA : ACM, 2006, p. 1283–1292 (cf. p. 80).

[3] Fraser Anderson et Walter F Bischof. « Learning and Performance with Gesture Guides ». In : *CHI '13*. New York, NY, USA : ACM, 2013, p. 1109–1118 (cf. p. 33, 37, 126).

[4] Georg Apitz et François Guimbretière. « CrossY: a crossing-based drawing application ». In : *UIST '04*. New York, New York, USA : ACM Press, oct. 2004, p. 3–12 (cf. p. 59, 60).

[5] Georg Apitz, François Guimbretière et Shumin Zhai. « Foundations for Designing and Evaluating User Interfaces Based on the Crossing Paradigm ». In : *ACM Trans. Comput.-Hum. Interact.* 17.2 (mai 2008), 9 :1–9 :42 (cf. p. 59, 60).

[6] Caroline Appert et Olivier Bau. « Scale detection for a priori gesture recognition ». In : *CHI '10*. New York, New York, USA : ACM Press, avr. 2010, p. 879 (cf. p. 27, 61).

[7] Caroline Appert et Shumin Zhai. « Using strokes as command shortcuts: cognitive benefits and toolkit support ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 2289–2298 (cf. p. 27, 31, 33, 36, 38, 111, 116, 137).

[8] Apple Computer Inc. « Macintosh Human Interface Guidelines ». USA : Addison-Wesley Publishing Company, 1992 (cf. p. 49, 50, 65).

[9] Ahmed Sabbir Arif et Wolfgang Stuerzlinger. « Pseudo-pressure Detection and Its Use in Predictive Text Entry on Touchscreens ». In : *OZCHI '13*. New York, NY, USA : ACM, 2013, p. 383–392 (cf. p. 18, 19).

[10] Oscar Kin-Chung Au et Chiew-Lan Tai. « Multitouch Finger Registration and Its Applications ». In : *OZCHI '10*. New York, New York, USA : ACM, nov. 2010, p. 41–48 (cf. p. 62, 68, 71, 72, 129–132).

[11] Mathias Baglioni. « Nouvelles interactions physiques pour dispositifs mobiles ». Thèse de doct. Paris, ENST, 2012 (cf. p. 28).

[12] Mathias Baglioni, Eric Lecolinet et Yves Guiard. « JerkTilts: Using Accelerometers for Eight-choice Selection on Mobile Devices ». In : *Proceedings of the 13th International Conference on Multimodal Interfaces*. New York, NY, USA : ACM, 2011, p. 121–128 (cf. p. 63).

[13] Mathias Baglioni, Sylvain Malacria, Eric Lecolinet et Yves Guiard. « Flick-and-brake: Finger Control over Inertial/Sustained Scroll Motion ». In : *CHI EA '11*. New York, NY, USA : ACM, 2011, p. 2281–2286 (cf. p. 18).

[14] Gilles Bailly. « Techniques de menus : Caractérisation, Conception et Evaluation ». Thèse de doct. Université Joseph-Fourier - Grenoble I, 2009 (cf. p. 35, 116).

[15] Gilles Bailly, Alexandre Demeure, Eric Lecolinet et Laurence Nigay. « MultiTouch menu (MTM) ». In : *IHM '08*. New York, New York, USA : ACM, sept. 2008, p. 165–168 (cf. p. 64, 74, 130, 132).

[16] Gilles Bailly, Eric Lecolinet et Yves Guiard. « Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces ». In : *CHI '10*. New York, New York, USA : ACM, avr. 2010, p. 591–594 (cf. p. 36, 48, 122, 131).

[17] Gilles Bailly, Eric Lecolinet et Laurence Nigay. « Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization ». In : *AVI '08*. New York, NY, USA : ACM, 2008, p. 15–22 (cf. p. 35, 36, 66, 112, 116).

[18] Gilles Bailly, Eric Lecolinet et Laurence Nigay. « Quinze ans de recherche sur les menus: critères et propriétés des techniques de menus ». In : *IHM '07*. New York, New York, USA : ACM Press, nov. 2007, p. 119–126 (cf. p. 35).

[19] Gilles Bailly, Eric Lecolinet et Laurence Nigay. « Wave menus: improving the novice mode of hierarchical marking menus ». In : *Interact '07*. Rio de Janeiro, Brazil : Springer, 2007, p. 475–488 (cf. p. 35).

[20] Gilles Bailly et Sylvain Malacria. « MenuInspector: Outil de l'analyse des menus et cas d'étude ». In : *IHM '13*. Bordeaux, France : ACM, 2013, p. 103–106 (cf. p. 6).

[21] Gilles Bailly, Jörg Müller et Eric Lecolinet. « Design and Evaluation of Finger-count Interaction: Combining Multitouch Gestures and Menus ». In : *IJHCS* 70.10 (oct. 2012), p. 673–689 (cf. p. 36, 48, 62, 122, 126, 131, 154).

[22] Gilles Bailly, Anne Roudaut, Eric Lecolinet et Laurence Nigay. « Menus Leaf: Enrichir Les Menus Linéaires Par Des Gestes ». In : *IHM '08*. New York, NY, USA : ACM, 2008, p. 169–172 (cf. p. 36).

[23] Ravin Balakrishnan et Pranay Patel. « The PadMouse: facilitating selection and spatial positioning for the non-dominant hand ». In : *CHI '98*. New York, New York, USA : ACM, jan. 1998, p. 9–16 (cf. p. 103).

[24] Nikola Banovic, Frank Chun Yat Li, David Dearman, Koji Yatani et Khai N. Truong. « Design of unimanual multi-finger pie menu interaction ». In : *ITS '11*. New York, New York, USA : ACM, nov. 2011, p. 120–129 (cf. p. 62, 63).

[25] Olivier Bau et Wendy Mackay. « OctoPocus: a dynamic guide for learning gesture-based command sets ». In : *UIST '08*. New York, New York, USA : ACM Press, oct. 2008, p. 37 (cf. p. 34, 35, 117).

[26] Olivier Bau, Ivan Poupyrev, Ali Israr et Chris Harrison. « TeslaTouch: Electrovibration for Touch Surfaces ». In : *UIST '10*. New York, NY, USA : ACM, 2010, p. 283–292 (cf. p. 23).

[27] Thomas Baudel et Michel Beaudouin-Lafon. « CHARADE: Remote Control of Objects using FreeHand Gestures ». In : *Communications of the ACM* 36 (1993), p. 28–35 (cf. p. 64).

[28] Patrick Baudisch, Torsten Becker et Frederik Rudeck. « Lumino: tangible blocks for tabletop computers based on glass fiber bundles ». In : *CHI '10*. New York, NY, USA : ACM, 2010, p. 1165–1174 (cf. p. 18).

[29] Michel Beaudouin-Lafon. « Instrumental interaction: an interaction model for designing post-WIMP user interfaces ». In : *CHI '00*. New York, NY, USA : ACM, 2000, p. 446–453 (cf. p. 20, 21).

[30] Michel Beaudouin-Lafon et Wendy Mackay. « Reification, polymorphism and reuse: three principles for designing visual interfaces ». In : *AVI '00*. New York, NY, USA : ACM, 2000, p. 102–109 (cf. p. 20, 21).

[31] Hrvoje Benko, Edward W Ishak et Steven Feiner. « Cross-dimensional gestural interaction techniques for hybrid immersive environments ». In : *VR '05*. 2005, p. 209–216 (cf. p. 133).

[32] Hrvoje Benko, T Scott Saponas, Dan Morris et Desney Tan. « Enhancing Input on and Above the Interactive Surface with Muscle Sensing ». In : *ITC*. New York, NY, USA : ACM, 2009, p. 93–100 (cf. p. 65, 129, 131, 133).

[33] Hrvoje Benko, Andrew D Wilson et Patrick Baudisch. « Precise Selection Techniques for Multi-touch Screens ». In : *CHI '06*. New York, NY, USA : ACM, 2006, p. 1263–1272 (cf. p. 18, 22, 130, 131).

[34] Mike Bennett, Kevin McCarthy, Sile O'Modhrain et Barry Smyth. « Simpleflow: Enhancing Gestural Interaction with Gesture Prediction, Abbreviation and Autocompletion ». In : *INTERACT '11*. Berlin, Heidelberg : Springer-Verlag, 2011, p. 591–608 (cf. p. 34).

[35] Eric A Bier, Maureen C Stone, Ken Fishkin, William Buxton et Thomas Baudel. « A Taxonomy of See-through Tools ». In : *CHI '94*. New York, NY, USA : ACM, 1994, p. 358–364 (cf. p. 58).

[36] Eric A Bier, Maureen C Stone, Ken Pier, Ken Fishkin, Thomas Baudel, Matt Conway, William Buxton et Tony DeRose. « Toolglass and magic lenses: the see-through interface ». In : *CHI '94*. New York, NY, USA : ACM, 1994, p. 445–446 (cf. p. 58).

[37] Richard A. Bolt. « "Put-that-there": Voice and gesture at the graphics interface ». In : *SIGGRAPH '80*. T. 14. 3. ACM, juil. 1980, p. 262–270 (cf. p. 64).

[38] David Bonnet, Caroline Appert et Michel Beaudouin-Lafon. « Extending the Vocabulary of Touch Events with ThumbRock ». In : *GI '13*. Toronto, Ont., Canada, Canada : Canadian Information Processing Society, 2013, p. 221–228 (cf. p. 61).

[39] Andrew Bragdon, Eugene Nelson, Yang Li et Ken Hinckley. « Experimental Analysis of Touch-screen Gesture Designs in Mobile Environments ». In : *CHI '11*. New York, NY, USA : ACM, 2011, p. 403–412 (cf. p. 59, 60).

[40] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller et Joseph J. LaViola. « GestureBar: improving the approachability of gesture-based interfaces ». In : *CHI '09*. New York, New York, USA : ACM Press, avr. 2009, p. 2269–2278 (cf. p. 33).

[41] Robert W Brainard, Thomas S Irby, Paul M Fitts et Earl A Alluisi. « Some variables influencing the rate of gain of information ». In : *Journal of Experimental Psychology* 63.2 (1962), p. 105–110 (cf. p. 113).

[42] Peter Brandl, Clifton Forlines, Daniel Wigdor, Michael Haller et Chia Shen. « Combining and Measuring the Benefits of Bimanual Pen and Direct-touch Interaction on Horizontal Interfaces ». In : *AVI '08*. New York, NY, USA : ACM, 2008, p. 154–161 (cf. p. 33).

[43] Stephen A Brewster, Peter C Wright et Alistair D N Edwards. « An Evaluation of Earcons for Use in Auditory Human-computer Interfaces ». In : *INTERACT '93 and CHI '93*. New York, NY, USA : ACM, 1993, p. 222–227 (cf. p. 52).

[44] William Buxton. « A three-state model of graphical input ». In : *Interact '90*. North-Holland Publishing Co., août 1990, p. 449–456 (cf. p. 13, 23, 25, 26, 47, 48, 132).

[45] William Buxton. « Chunking and phrasing and the design of human-computer dialogues ». In : *Human-computer interaction : toward the year 2000*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1995, p. 494–499 (cf. p. 56).

[46] William Buxton. « Touch, Gesture & Marking ». In : *Readings in Human Computer Interaction : Toward the Year 2000*. Morgan Kaufmann, 1995. Chap. 7 (cf. p. 27).

[47] William Buxton, Eugene Fiume, Ralph Hill, Alison Lee et Carson Woo. « Continuous hand-gesture driven input ». In : *GI '83*. Mai 1983, p. 191–195 (cf. p. 60).

[48] William Buxton, Ralph Hill et Peter Rowley. « Issues and Techniques in Touch-sensitive Tablet Input ». In : *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : ACM, 1985, p. 215–224 (cf. p. 18).

[49] Xiang Cao, A D Wilson, R Balakrishnan, K Hinckley et S E Hudson. « ShapeTouch: Leveraging contact shape on interactive surfaces ». In : *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*. Oct. 2008, p. 129–136 (cf. p. 64).

[50] Xiang Cao et Shumin Zhai. « Modeling human performance of pen stroke gestures ». In : *CHI '07*. New York, New York, USA : ACM Press, avr. 2007, p. 1495 (cf. p. 31).

[51] Stuart K Card, Thomas P Moran et Allen Newell. « The Keystroke-level Model for User Performance Time with Interactive Systems ». In : *Commun. ACM* 23.7 (juil. 1980), p. 396–410 (cf. p. 27).

[52] John M Carroll et Mary Beth Rosson. « Paradox of the active user. » The MIT Press, 1987 (cf. p. 37, 111).

[53] Géry Casiez et Nicolas Roussel. « No More Bricolage!: Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions ». In : *UIST '11*. New York, NY, USA : ACM, 2011, p. 603–614 (cf. p. 24).

[54] Géry Casiez, Nicolas Roussel, Romuald Vanbelleghem et Frédéric Giraud. « Surfpad: Riding Towards Targets on a Squeeze Film Effect ». In : *CHI '11*. New York, NY, USA : ACM, 2011, p. 2491–2500 (cf. p. 23).

[55] Steven J. Castellucci et Scott MacKenzie. « Graffiti vs. unistrokes: An Empirical Comparison ». In : *CHI '08*. New York, New York, USA : ACM Press, avr. 2008, p. 305 (cf. p. 114).

[56] Liwei Chan, Stefanie Müller, Anne Roudaut et Patrick Baudisch. « CapStones and ZebraWidgets: Sensing Stacks of Building Blocks, Dials and Sliders on Capacitive Touch Screens ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 2189–2192 (cf. p. 15, 17).

[57] Nicholas Y. Chen, François Guimbretière et Corinna Löckenhoff. « Relative role of merging and two-handed operation on command selection speed ». In : *IJHCS* 66.10 (oct. 2008), p. 729–740 (cf. p. 36, 126).

[58] Andy Cockburn, David Ahlström et Carl Gutwin. « Understanding Performance in Touch Selections: Tap, Drag and Radial Pointing Drag with Finger, Stylus and Mouse ». In : *Int. J. Hum.-Comput. Stud.* 70.3 (mar. 2012), p. 218–233 (cf. p. 25).

[59] Andy Cockburn, Carl Gutwin, Joey Scarr et Sylvain Malacria. « Supporting Novice to Expert Transitions in User Interfaces ». In : *ACM Comput. Surv.* 47.2 (nov. 2014), 31 :1–31 :36 (cf. p. 26, 27, 37, 67, 69, 76, 111, 112, 123).

[60] Andy Cockburn, Per Ola Kristensson, Jason Alexander et Shumin Zhai. « Hard Lessons: Effort-inducing Interfaces Benefit Spatial Learning ». In : *CHI '07*. New York, NY, USA : ACM, 2007, p. 1571–1580 (cf. p. 37).

[61] Aurélie Cohé. « Manipulation de contenu 3D sur des surfaces tactiles ». Thèse de doct. 2012 (cf. p. 14).

[62] Philip R Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen et Josh Clow. « QuickSet: Multimodal Interaction for Distributed Applications ». In : *Proceedings of the Fifth ACM International Conference on Multimedia*. New York, NY, USA : ACM, 1997, p. 31–40 (cf. p. 64).

[63] Commission Générale de Terminologie et de Néologie. « Vocabulaire de l'audiovisuel et de l'informatique ». In : *Journal Officiel de la République Française* 0043 (2011), p. 3238 (cf. p. 11).

[64] Sashikanth Damaraju, Jinsil Hwaryoung Seo, Tracy Hammond et Andruid Kerne. « Multi-tap sliders: advancing touch interaction for parameter adjustment ». In : *IUI '13*. 2013, p. 445–452 (cf. p. 61, 62, 126).

[65] Paul A David. « Clio and the Economics of QWERTY ». In : *American Economic Review* 75.2 (mai 1985), p. 332–337 (cf. p. 38, 121–123).

[66] Paul Dietz et Darren Leigh. « DiamondTouch: a multi-user touch technology ». In : *UIST '01*. New York, New York, USA : ACM Press, 2001, p. 219 (cf. p. 18).

[67] Andre Doucette, Carl Gutwin et Regan L. Mandryk. « A comparison of techniques for in-place toolbars ». In : *GI '10*. Canadian Information Processing Society, mai 2010, p. 35–38 (cf. p. 60).

[68] J G Elias, W C Westerman et M M Haggerty. « Multi-touch gesture dictionary ». 2007 (cf. p. 33).

[69] Philipp Ewerling, Alexander Kulik et Bernd Froehlich. « Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions ». In : *ITS '12*. New York, New York, USA : ACM Press, nov. 2012, p. 173–182 (cf. p. 71, 129, 131, 133).

[70] Paul M Fitts. « The information capacity of the human motor system in controlling the amplitude of movement ». In : *Journal of Experimental PSychology* 74 (1954), p. 381–391 (cf. p. 25, 130, 138).

[71] George Fitzmaurice, Azam Khan, Robert Pieké, William Buxton et Gordon Kurtenbach. « Tracking Menus ». In : *UIST '03*. New York, New York, USA : ACM, nov. 2003, p. 71–79 (cf. p. 65).

[72] George Fitzmaurice, Justin Matejka, Azam Khan, Mike Glueck et Gordon Kurtenbach. « PieCursor: Merging Pointing and Command Selection for Rapid In-place Tool Switching ». In : *CHI '08*. New York, New York, USA : ACM Press, avr. 2008, p. 1361 (cf. p. 65).

[73] Clifton Forlines, Daniel Wigdor, Chia Shen et Ravin Balakrishnan. « Direct-touch vs. Mouse Input for Tabletop Displays ». In : *CHI '07*. New York, NY, USA : ACM, 2007, p. 647–656 (cf. p. 25).

[74] Cédric Foucault, Manfred Micaux, David Bonnet et Michel Beaudouin-Lafon. « SPad: A Bimanual Interaction Technique for Productivity Applications on Multi-touch Tablets ». In : *CHI EA '14*. New York, NY, USA : ACM, 2014, p. 1879–1884 (cf. p. 6, 52, 58).

[75] Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran et Laurence Nigay. « Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode ». In : *AVI '00*. New York, NY, USA : ACM, 2010, p. 173–180 (cf. p. 35, 36, 88).

[76] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris et Daniel Wigdor. « ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures ». In : *ITS '09*. New York, New York, USA : ACM, nov. 2009, p. 165–172 (cf. p. 18, 33–35, 55–57, 64, 81).

[77] Gennadi Blindmann. « Multitouch Solution » (cf. p. 14).

[78] Don Gentner et Jakob Nielsen. « The Anti-Mac Interface ». In : *Commun. ACM* 39.8 (août 1996), p. 70–82 (cf. p. 53).

[79] Emilien Ghomi, Stéphane Huot, Olivier Bau, Michel Beaudouin-Lafon et Wendy Mackay. « Arpège: Learning Multitouch Chord Gestures Vocabularies ». In : *ITS '13*. New York, NY, USA : ACM, 2013, p. 209–218 (cf. p. 34, 132).

[80] Jérémie Gilliot. « Interactions multi-points indirectes sur grands écrans ». Thèse de doct. 2014 (cf. p. 14, 24).

[81] Jérémie Gilliot, Géry Casiez et Nicolas Roussel. « Impact of Form Factors and Input Conditions on Absolute Indirect-touch Pointing Tasks ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 723–732 (cf. p. 12).

[82] Alix Goguey, Géry Casiez, Thomas Pietrzak, Daniel Vogel et Nicolas Roussel. « Adoiraccourcix: Multi-touch Command Selection Using Finger Identification ». In : *IHM '14*. New York, NY, USA : ACM, 2014, p. 28–37 (cf. p. 129–133).

[83] Alix Goguey, Géry Casiez, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak et Nicolas Roussel. « A Three-step Interaction Pattern for Improving Discoverability in Finger Identification Techniques ». In : *UIST '14 Adjunct*. New York, NY, USA : ACM, 2014, p. 33–34 (cf. p. 129, 131–133).

[84] John D Gould et Josiane Salaun. « Behavioral Experiments on Handmarkings ». In : *SIGCHI Bull.* 18.4 (mai 1986), p. 175–181 (cf. p. 31, 32).

[85] Tovi Grossman, Pierre Dragicevic et Ravin Balakrishnan. « Strategies for accelerating on-line learning of hotkeys ». In : *CHI '07*. New York, NY, USA : ACM, 2007, p. 1591–1600 (cf. p. 27, 111).

[86] Tovi Grossman, Ken Hinckley, Patrick Baudisch, Maneesh Agrawala et Ravin Balakrishnan. « Hover widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices ». In : *CHI '06*. New York, New York, USA : ACM Press, avr. 2006, p. 861–870 (cf. p. 65).

[87] Yves Guiard. « On Fitts's and Hooke's laws: Simple harmonic movement in upper-limb cyclical aiming ». In : *Acta Psychologica* 82.1-3 (mar. 1993), p. 139–159 (cf. p. 25).

[88] François Guimbretière, Andrew Martin et Terry Winograd. « Benefits of Merging Command Selection and Direct Manipulation ». In : *ACM ToCHI* 12.3 (sept. 2005), p. 460–476 (cf. p. 56, 59, 67, 77, 126).

[89] François Guimbretière, Maureen Stone et Terry Winograd. « Fluid Interaction with High-resolution Wall-size Displays ». In : *UIST '01*. New York, NY, USA : ACM, 2001, p. 21–30 (cf. p. 60).

[90] François Guimbretière et Terry Winograd. « FlowMenu: combining command, text, and data entry ». In : *UIST '00*. New York, New York, USA : ACM, nov. 2000, p. 213–216 (cf. p. 35, 36, 56, 59, 60, 67, 77, 84, 107, 126).

[91] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria et Scott Olson. « Faster Command Selection on Tablets with FastTap ». In : *CHI '14*. Toronto : ACM Press, 2014 (cf. p. 58).

[92] Peter A Hancock et Karl M Newell. « The movement speed-accuracy relationship in space-time ». In : *Motor Behavior*. Springer, 1985, p. 153–188 (cf. p. 139).

[93] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon et Roy Want. « Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces ». In : *CHI '98*. New York, New York, USA : ACM Press, jan. 1998, p. 17–24 (cf. p. 138).

[94] Chris Harrison, Brian Amento, Stacey Kuznetsov et Robert Bell. « Rethinking the progress bar ». In : *UIST '07*. New York, New York, USA : ACM Press, oct. 2007, p. 115 (cf. p. 62, 63).

[95] Chris Harrison, Julia Schwarz et Scott E Hudson. « TapSense: Enhancing Finger Interaction on Touch Surfaces ». In : *UIST '11*. New York, NY, USA : ACM, 2011, p. 627–636 (cf. p. 131).

[96] Chris Harrison, Robert Xiao, Julia Schwarz et Scott E Hudson. « TouchTools: Leveraging Familiarity and Skill with Physical Tools to Augment Touch Interaction ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 2913–2916 (cf. p. 56).

[97] Christopher F. Herot et Guy Weinzapfel. « One-point touch input of vector information for computer displays ». In : *SIGGRAPH '78*. T. 12. 3. New York, NY, USA : ACM, 1978, p. 210–216 (cf. p. 18, 131).

[98] William E Hick. « On the rate of gain of information ». In : *Quarterly Journal of Experimental Psychology* 4.1 (1952), p. 11–26 (cf. p. 53, 138).

[99] Ken Hinckley, Patrick Baudisch, Gonzalo Ramos et François Guimbretière. « Design and analysis of delimiters for selection-action pen gesture phrases in Scriboli ». In : *CHI '05*. New York, NY, USA : ACM, 2005, p. 451–460 (cf. p. 46, 47, 54, 57, 58, 60, 61, 80).

[100] Ken Hinckley, François Guimbretière, Maneesh Agrawala, Georg Apitz et Nicholas Chen. « Phrasing techniques for multi-stroke selection gestures ». In : *GI '06*. 2006, p. 147–154 (cf. p. 57, 60).

[101] Ken Hinckley, François Guimbretière, Patrick Baudisch, Raman Sarin, Maneesh Agrawala et Ed Cutrell. « The Springboard: Multiple Modes in One Spring-Loaded Control ». In : *CHI '06*. New York, New York, USA : ACM Press, avr. 2006, p. 181 (cf. p. 46, 48, 49, 52, 65, 155).

[102] Ken Hinckley et Mike Sinclair. « Touch-sensing Input Devices ». In : *CHI '99*. New York, NY, USA : ACM, 1999, p. 223–230 (cf. p. 12, 23).

[103] Ken Hinckley et Hyunyoung Song. « Sensor Synaesthesia: Touch in Motion, and Motion in Touch ». In : *CHI '11*. New York, NY, USA : ACM, 2011, p. 801–810 (cf. p. 61, 63, 131).

[104] Steve Hodges, Shahram Izadi, Alex Butler, Alban Rrustemi et William Buxton. « ThinSight: Versatile Multi-touch Sensing for Thin Form-factor Displays ». In : *UIST '07*. New York, NY, USA : ACM, 2007, p. 259–268 (cf. p. 16).

[105] Richard Holcomb et Alan L. Tharp. « The Effect of Windows on Man-machine Interfaces (or Opening Doors with Windows) ». In : *Proceedings of the 1985 ACM Thirteenth Annual Conference on Computer Science*. New York, New York, USA : ACM Press, mar. 1985, p. 280–291 (cf. p. 51, 58, 59).

[106] Christian Holz et Patrick Baudisch. « Fiberio: a touchscreen that senses fingerprints ». In : *UIST '13*. New York, New York, USA : ACM Press, oct. 2013, p. 41–50 (cf. p. 18, 129–131, 133).

[107] Christian Holz et Patrick Baudisch. « The Generalized Perceived Input Point Model and How to Double Touch Accuracy by Extracting Fingerprints ». In : *CHI '10*. New York, NY, USA : ACM, 2010, p. 581–590 (cf. p. 21, 22, 133).

[108] Stéphane Huot et Eric Lecolinet. « ArchMenu Et ThumbMenu: Contrôler Son Dispositif Mobile « Sur Le Pouce » ». In : *IHM '07*. New York, NY, USA : ACM, 2007, p. 107–110 (cf. p. 36).

[109] Edwin L Hutchins, James D Hollan et Donald A Norman. « Direct Manipulation Interfaces ». In : *Hum.-Comput. Interact.* 1.4 (déc. 1985), p. 311–338 (cf. p. 19, 20).

[110] Ray Hyman. « Stimulus information as a determinant of reaction time. » In : *Journal of Experimental Psychology* 45.3 (1953), p. 188–196 (cf. p. 113, 121).

[111] Intel Free Press. « How Touch Computing Works ». 2012 (cf. p. 14).

[112] Poika Isokoski. « Model for Unistroke Writing Time ». In : *CHI '01*. New York, NY, USA : ACM, 2001, p. 357–364 (cf. p. 31).

[113] Mohit Jain et Ravin Balakrishnan. « User Learning and Performance with Bezel Menus ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 2221–2230 (cf. p. 59).

[114] Herbert D Jellinek et Stuart K Card. « Powermice and User Performance ». In : *CHI '90*. New York, NY, USA : ACM, 1990, p. 213–220 (cf. p. 25).

[115] Gabe Johnson, Mark D Gross et Ellen Yi-Luen Do. « Flow selection: a time-based selection and operation technique for sketching tools ». In : *AVI '06*. New York, New York, USA : ACM Press, mai 2006, p. 83 (cf. p. 58, 60).

[116] Jeff Johnson. « Modes in non-computer devices ». In : *International Journal of Man-Machine Studies* 32.4 (avr. 1990), p. 423–438 (cf. p. 41, 42, 45, 48, 50–53).

[117] Jeff Johnson et G. Engelbeck. « Modes survey results ». In : *ACM SIGCHI Bulletin* 20.4 (avr. 1989), p. 38–50 (cf. p. 42, 46).

[118] Gregory V Jones. « Tests of the dual-mechanism theory of recall ». In : *Acta Psychologica* 50.1 (1982), p. 61–72 (cf. p. 117).

[119] Maria Karam et M C Schraefel. « A taxonomy of gestures in human computer interactions ». Rapp. tech. University of Southampton, 2005 (cf. p. 28).

[120] David Kieras, David Meyer et James Ballas. « Towards Demystification of Direct Manipulation: Cognitive Modeling Charts the Gulf of Execution ». In : *CHI '01*. New York, NY, USA : ACM, 2001, p. 128–135 (cf. p. 19).

[121] Sven Kratz, Tilo Westermann, Michael Rohs et Georg Essl. « CapWidgets: tangile widgets versus multi-touch controls on mobile devices ». In : *CHI '11 Extended Abstracts*. Vancouver, BC, Canada, 2011, p. 1351–1356 (cf. p. 15, 17).

[122] Per Ola Kristensson et L C Denby. « Continuous Recognition and Visualization of Pen Strokes and Touch-screen Gestures ». In : *SBIM '11*. New York, NY, USA : ACM, 2011, p. 95–102 (cf. p. 34).

[123] Per Ola Kristensson et Shumin Zhai. « Command strokes with and without preview: using pen gestures on keyboard for command selection ». In : *CHI '07*. New York, New York, USA : ACM Press, avr. 2007, p. 1137–1146 (cf. p. 33).

[124] Peter Kung, Dominik Küser, Craig Schroeder, Tony DeRose, Donald Greenberg et Kenrick Kin. « An augmented multi-touch system using hand and finger identification ». In : *CHI EA '12*. New York, NY, USA : ACM, 2012, p. 1431–1432 (cf. p. 133).

[125] Gordon Kurtenbach. « The Design and Evaluation of Marking Menus ». Thèse de doct. 1993 (cf. p. 34–36, 55, 56, 61, 66).

[126] Gordon Kurtenbach et William Buxton. « Issues in combining marking and direct manipulation techniques ». In : *UIST '91*. New York, New York, USA : ACM, oct. 1991, p. 137–144 (cf. p. 27, 34–36, 41, 49, 55, 56, 58, 60, 66, 76, 111, 112, 116, 131).

[127] Gordon Kurtenbach et William Buxton. « The limits of expert performance using hierarchic marking menus ». In : *INTERACT '93 and CHI '93*. New York, NY, USA : ACM, 1993, p. 482–487 (cf. p. 34–36, 60, 76, 112).

[128] Gordon Kurtenbach et William Buxton. « User learning and performance with marking menus ». In : *CHI '94*. New York, NY, USA : ACM, 1994, p. 258–264 (cf. p. 34, 36, 37, 67, 69, 76, 111–113, 116).

[129] Gordon Kurtenbach, George W. Fitzmaurice, Russell N. Owen et Thomas Baudel. « The Hotbox: Efficient Access to a Large Number of Menu-items ». In : *CHI '99*. New York, New York, USA : ACM Press, mai 1999, p. 231–237 (cf. p. 58).

[130] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel et William Buxton. « The Design of a GUI Paradigm Based on Tablets, Two-hands, and Transparency ». In : *CHI '97*. New York, NY, USA : ACM, 1997, p. 35–42 (cf. p. 58).

[131] Gordon Kurtenbach, George Fitzmaurice, Azam Khan et Don Almeida. « Scale Independence in Marking Menus ». 2004 (cf. p. 56).

[132] Gordon Kurtenbach, Thomas P Moran et William Buxton. « Contextual Animation of Gestural Commands ». In : *Computer Graphics Forum* 13.5 (1994), p. 305–314 (cf. p. 33–37, 67, 69, 76, 112, 115).

[133] Gordon Kurtenbach, Abigail Sellen et William Buxton. « An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus ». In : *Human-Computer Interaction* 8.1 (mar. 1993), p. 1–23 (cf. p. 34–36, 117).

[134] James A Landay. « SILK: Sketching Interfaces Like Krazy ». In : *Conference Companion on Human Factors in Computing Systems*. New York, NY, USA : ACM, 1996, p. 398–399 (cf. p. 60).

[135] David M Lane, H Albert Napier, S Camille Peres et Aniko Sandor. « Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts ». In : *Int. J. Hum. Comput. Interaction* (2005), p. 133–144 (cf. p. 37).

[136] Bhoram Lee, Hyunjeong Lee, Soo-Chul Lim, Hyungkew Lee, Seungju Han et Joonah Park. « Evaluation of human tangential force input performance ». In : *CHI '12*. 2012, p. 3121–3130 (cf. p. 19).

[137] S K Lee, William Buxton et K C Smith. « A Multi-touch Three Dimensional Touch-sensitive Tablet ». In : *CHI '85*. New York, NY, USA : ACM, 1985, p. 21–25 (cf. p. 131).

[138] Seungyon Lee et Shumin Zhai. « The Performance of Touch Screen Soft Buttons ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 309–318 (cf. p. 25, 26).

[139] Julian Lepinski, Tovi Grossman et George Fitzmaurice. « The design and evaluation of multitouch marking menus ». In : *CHI '10*. New York, New York, USA : ACM, avr. 2010, p. 2233–2242 (cf. p. 35, 36, 62, 65, 129, 131, 133).

[140] Yang Li. « Gesture search: A Tool for Fast Mobile Data Access ». In : *UIST '10*. New York, New York, USA : ACM Press, oct. 2010, p. 87 (cf. p. 114).

[141] Yang Li, Ken Hinckley, Zhiwei Guan et James A Landay. « Experimental Analysis of Mode Switching Techniques in Pen-Based User Interfaces ». In : *CHI '05*. New York, New York, USA : ACM, avr. 2005, p. 461–470 (cf. p. 46, 54, 55, 60, 65, 87).

[142] I Scott MacKenzie. « Fitts' Law As a Research and Design Tool in Human-computer Interaction ». In : *Hum.-Comput. Interact.* 7.1 (mar. 1992), p. 91–139 (cf. p. 25).

[143] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn et Carl Gutwin. « Promoting Hotkey Use Through Rehearsal with ExposeHK ». In : *CHI '13*. New York, NY, USA : ACM, 2013, p. 573–582 (cf. p. 37, 111).

[144] Sylvain Malacria, Eric Lecolinet et Yves Guiard. « Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: the cyclostar approach ». In : *CHI '10*. New York, NY, USA : ACM, avr. 2010, p. 2615–2624 (cf. p. 56, 61, 81, 90).

[145] Marcel Brown. « The "Lost" Steve Jobs Speech from 1983; Foreshadowing Wireless Networking, the iPad, and the App Store ». 2012 (cf. p. 11).

[146] Anders Markussen, Mikkel Rønne Jakobsen et Kasper Hornbæk. « Vulture: A Mid-air Word-gesture Keyboard ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 1073–1082 (cf. p. 65).

[147] Nicolai Marquardt, Johannes Kiemer et Saul Greenberg. « What Caused That Touch?: Expressive Interaction with a Surface Through Fiduciary-tagged Gloves ». In : *ITS '10*. New York, NY, USA : ACM, 2010, p. 139–142 (cf. p. 65, 129, 131, 133).

[148] Anthony Martinet. « Etude de l'infuence de la séparation des degrés de liberté pour la manipulation 3-D à l'aide de surfaces tactiles multipoints ». Thèse de doct. 2011 (cf. p. 14).

[149] Justin Matejka, Tovi Grossman, Jessica Lo et George Fitzmaurice. « The design and evaluation of multi-finger mouse emulation techniques ». In : *CHI '09*. New York, New York, USA : ACM Press, avr. 2009, p. 1073 (cf. p. 62).

[150] David E Meyer, J E Keith-Smith, Sylvan Kornblum, Richard A Abrams et Charles E Wright. « Speed-accuracy tradeoffs in aimed movements : Toward a theory of rapid voluntary action. » In : (1990) (cf. p. 139).

[151] Norman Meyrowitz et Andries van Dam. « Interactive Editing Systems: Part II ». In : *ACM Comput. Surv.* 14.3 (sept. 1982), p. 353–415 (cf. p. 49).

[152] George A Miller. « The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information ». In : *The Psychological Review* 63.2 (1956), p. 81–97 (cf. p. 138).

[153] Sachi Mizobuchi, Shinya Terasaki, Turo Keski-Jaskari, Jari Nousiainen, Matti Ryynanen et Miika Silfverberg. « Making an Impression: Force-controlled Pen Input for Handheld Devices ». In : *CHI EA '05*. New York, NY, USA : ACM, 2005, p. 1661–1664 (cf. p. 18, 131).

[154] Andrew Monk. « Mode errors: a user-centred analysis and some preventative measures using keying-contingent sound ». In : *International Journal of Man-Machine Studies* 24.4 (avr. 1986), p. 313–327 (cf. p. 51–53).

[155] Denis Mottet, Reinoud J Bootsma, Yves Guiard et Michel Laurent. « Fitts' law in two-dimensional task space ». In : *Experimental brain research* 100.1 (1994), p. 144–148 (cf. p. 25).

[156] Michael Moyle et Andy Cockburn. « The Design and Evaluation of a Flick Gesture for 'Back' and 'Forward' in Web Browsers. » In : *AUIC '02*. T. 18. Australian Computer Society, 2003, p. 39–46 (cf. p. 62).

[157] Elizabeth D Mynatt, Takeo Igarashi, W Keith Edwards et Anthony LaMarca. « Flatland: New Dimensions in Office Whiteboards ». In : *CHI '99*. New York, NY, USA : ACM, 1999, p. 346–353 (cf. p. 60).

[158] Mathieu Nancel, Daniel Vogel et Edward Lank. « Clutching Is Not (Necessarily) the Enemy ». In : *CHI '15*. New York, NY, USA : ACM, 2015, p. 4199–4202 (cf. p. 25).

[159] Donald A Norman. « Categorization of action slips ». In : *Psychological Review* 88.1 (1981), p. 1–15 (cf. p. 26, 51).

[160] Donald A Norman. « Design Rules Based on Analyses of Human Error ». In : *Commun. ACM* 26.4 (avr. 1983), p. 254–258 (cf. p. 41, 51, 53).

[161] Donald A Norman. « Natural User Interfaces Are Not Natural ». In : *interactions* 17.3 (mai 2010), p. 6–10 (cf. p. 32).

[162] Donald A. Norman. « The Psychology of Everyday Things ». Basic Books, 1988 (cf. p. 20, 26).

[163] Donald A Norman et Jakob Nielsen. « Gestural Interfaces: A Step Backward in Usability ». In : *interactions* 17.5 (sept. 2010), p. 46–49 (cf. p. 32).

[164] Ian Oakley et Junseok Park. « A Motion-based Marking Menu System ». In : *CHI EA '07*. New York, NY, USA : ACM, 2007, p. 2597–2602 (cf. p. 35, 36).

[165] Daniel L. Odell, Richard C. Davis, Andrew Smith et Paul K. Wright. « Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques ». In : *GI '04*. Canadian Human-Computer Communications Society, mai 2004, p. 17–24 (cf. p. 35).

[166] Alex Olwal, Steven Feiner et Susanna Heyman. « Rubbing and tapping for precise and rapid selection on touch-screen displays ». In : *CHI '08*. New York, New York, USA : ACM, avr. 2008, p. 295–304 (cf. p. 56, 61).

[167] Réjean Plamondon et Adel M Alimi. « Speed/accuracy trade-offs in target-directed movements ». In : *Behavioral and Brain Sciences* 20.02 (1997), p. 279–303 (cf. p. 139).

[168] Merle F. Poller et Susan K. Garter. « A comparative study of moded and modeless text editing by experienced editor users ». In : *CHI '83*. New York, New York, USA : ACM Press, déc. 1983, p. 166–170 (cf. p. 42, 44, 45, 48, 51).

[169] Stuart Pook, Eric Lecolinet, Guy Vaysseix et Emmanuel Barillot. « Control Menus: Execution and Control in a Single Interactor ». In : *CHI '00*. New York, New York, USA : ACM, avr. 2000, p. 263–264 (cf. p. 35, 36, 49, 56, 57, 59, 66, 67, 77, 84, 107, 110, 126).

[170] Alfred Poor. « How it works: The technology of touch screens » (cf. p. 14).

[171] Richard L. Potter, Linda J. Weldon et Ben Shneiderman. « Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies ». In : *CHI '88*. New York, NY, USA : ACM, 1988, p. 27–32 (cf. p. 21, 91, 130).

[172] Mahfuz Rahman, Sean Gustafson, Pourang Irani et Sriram Subramanian. « Tilt Techniques: Investigating the Dexterity of Wrist-based Input ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 1943–1952 (cf. p. 63).

[173] Gonzalo Ramos, Matthew Boulos et Ravin Balakrishnan. « Pressure Widgets ». In : *CHI '04*. New York, NY, USA : ACM, 2004, p. 487–494 (cf. p. 18, 65, 131).

[174] Jef Raskin. « Intuitive Equals Familiar ». In : *Commun. ACM* 37.9 (sept. 1994), p. 17–18 (cf. p. 31).

[175] Jef Raskin. « The Humane Interface: New Directions for Designing Interactive Systems ». ACM Press, mar. 2000 (cf. p. 31, 42, 46, 49, 56).

[176] Jun Rekimoto. « SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces ». In : *CHI '02*. New York, NY, USA : ACM, 2002, p. 113–120 (cf. p. 15, 17).

[177] O Rioul. « Théorie de l'information et du codage ». Lavoisier, 2007 (cf. p. 140, 141).

[178] Ilya D Rosenberg, Alexander Grau, Charles Hendee, Nadim Awad et Ken Perlin. « IMPAD: An Inexpensive Multi-touchpressure Acquisition Device ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 3217–3222 (cf. p. 131).

[179] Volker Roth et Thea Turner. « Bezel Swipe: Conflict-free Scrolling and Multiple Selection on Mobile Touch Screen Devices ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 1523–1526 (cf. p. 59).

[180] Anne Roudaut, Mathias Baglioni et Eric Lecolinet. « TimeTilt: using sensor-based gestures to travel through multiple applications on a mobile device ». In : *Interact '09*. Uppsala, Sweden : Springer-Verlag, 2009, p. 830–834 (cf. p. 63).

[181] Anne Roudaut, Gilles Bailly, Eric Lecolinet et Laurence Nigay. « Leaf Menus: Linear Menus with Stroke Shortcuts for Small Handheld Devices ». In : *Interact '09*. Uppsala, Sweden : Springer-Verlag, 2009, p. 616–619 (cf. p. 36).

[182] Anne Roudaut, Stéphane Huot et Eric Lecolinet. « TapTap and MagStick: Improving One-handed Target Acquisition on Small Touch-screens ». In : *AVI '08*. New York, NY, USA : ACM, 2008, p. 146–153 (cf. p. 22, 130, 135).

[183] Anne Roudaut, Eric Lecolinet et Yves Guiard. « MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb ». In : *CHI '09*. New York, New York, USA : ACM, avr. 2009, p. 927–936 (cf. p. 61).

[184] Quentin Roy, Yves Guiard, Gilles Bailly, Éric Lecolinet et Olivier Rioul. « Glass+Skin: An Empirical Evaluation of the Added Value of Finger Identification to Basic Single-Touch Interaction on Touch Screens ». In : *Interact '15*. Bamberg, Germany : Springer International Publishing, 2015, p. 55–71 (cf. p. 130, 154).

[185] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet et James Eagan. « Augmented Letters: Mnemonic gesture-based shortcuts ». In : *CHI '13*. New York, New York, USA : ACM, avr. 2013, p. 2325–2328 (cf. p. 34, 35, 38, 113, 154).

[186] Jaime Ruiz, Andrea Bunt et Edward Lank. « A Model of Non-preferred Hand Mode Switching ». In : *GI '08*. Toronto, Ont., Canada, Canada : Canadian Information Processing Society, 2008, p. 49–56 (cf. p. 60).

[187] Jaime Ruiz et Yang Li. « DoubleFlip: a motion gesture delimiter for interaction ». In : *UIST '10*. New York, New York, USA : ACM, oct. 2010, p. 449–450 (cf. p. 63).

[188] Joseph D Rutledge et Ted Selker. « Force-to-motion Functions for Pointing ». In : *Interact '90*. Amsterdam, The Netherlands : North-Holland Publishing Co., 1990, p. 701–706 (cf. p. 12, 13, 24).

[189] Hokyoung Ryu. « Will it be upper-case or will it be lower-case: can a prompt for text be a mode signal? » In : *CHI '02*. New York, New York, USA : ACM Press, avr. 2002, p. 836 (cf. p. 43, 52).

[190] Nadine B Sarter et David D Woods. « How in the world did we ever get into that mode? Mode error and awareness in supervisory control ». In : *Human Factors : The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), p. 5–19 (cf. p. 50, 53).

[191] Farzan Sasangohar, I Scott Mackenzie et Stacey D Scott. « Evaluation of Mouse and Touch Input for a Tabletop Display Using Fitts' Reciprocal Tapping Task ». In : *HFES '09*. 2009, p. 839–843 (cf. p. 25).

[192] Ferdinand de Saussure. « Cours de linguistique générale ». 1916 (cf. p. 113).

[193] Joey Scarr, Andy Cockburn, Carl Gutwin et Andrea Bunt. « Improving command selection with CommandMaps ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 257–266 (cf. p. 113).

[194] Joey Scarr, Andy Cockburn, Carl Gutwin et Andrea Bunt. « Improving command selection with CommandMaps ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 257–266 (cf. p. 116, 117).

[195] Joey Scarr, Andy Cockburn, Carl Gutwin, Andrea Bunt et Jared E Cechanowicz. « The Usability of CommandMaps in Realistic Tasks ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 2241–2250 (cf. p. 117).

[196] Joey Scarr, Andy Cockburn, Carl Gutwin et Philip Quinn. « Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces ». In : *CHI '11*. New York, New York, USA : ACM Press, mai 2011, p. 2741–2750 (cf. p. 37, 111, 113, 123).

[197] Andrew Sears et Ben Shneiderman. « High Precision Touchscreens: Design Strategies and Comparisons with a Mouse ». In : *Int. J. Man-Mach. Stud.* 34.4 (avr. 1991), p. 593–613 (cf. p. 25).

[198] Abigail Sellen, Gordon Kurtenbach et William Buxton. « The prevention of mode errors through sensory feedback ». In : *Hum.-Comput. Interact.* 7.2 (1992), p. 141–164 (cf. p. 41, 46, 51, 52, 60).

[199] Marcos Serrano, Eric Lecolinet et Yves Guiard. « Bezel-Tap Gestures: Quick Activation of Commands from Sleep Mode on Tablets ». In : *CHI '13*. New York, New York, USA : ACM, avr. 2013, p. 3027–3036 (cf. p. 63).

[200] Claude E. Shannon. « A Mathematical Theory of Communication ». In : *The Bell System Technical Journal* 27 (1948), p. 379–423, 623–656 (cf. p. 9, 130, 138–140, 144, 154, 156).

[201] Ben Shneiderman. « Direct Manipulation: A Step Beyond Programming Languages ». In : *Computer* 16.8 (août 1983), p. 57–69 (cf. p. 19, 20, 38).

[202] Ben Shneiderman. « Touch screens now offer compelling uses ». In : *Software, IEEE* 8.2 (mar. 1991), p. 93–94 (cf. p. 20, 21).

[203] R W Soukoreff et I S MacKenzie. « An informatic rationale for the speed-accuracy trade-off ». In : *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. Oct. 2009, p. 2890–2896 (cf. p. 139).

[204] Martin Spindler, Martin Schuessler, Marcel Martsch et Raimund Dachselt. « Pinch-drag-flick vs. Spatial Input: Rethinking Zoom & Pan on Mobile Displays ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 1113–1122 (cf. p. 63, 64).

[205] Mandayam A Srinivasan et Jyh-shing Chen. « Human performance in controlling normal forces of contact with rigid objects ». In : *Advances in Robotics, Mechatronics, and Haptic Interfaces 1993* 49 (1993), p. 119–125 (cf. p. 18).

[206] Craig Stewart, Michael Rohs, Sven Kratz et Georg Essl. « Characteristics of Pressure-based Input for Mobile Devices ». In : *CHi '10*. New York, NY, USA : ACM, 2010, p. 801–810 (cf. p. 18).

[207] David J Sturman et David Zeltzer. « A Survey of Glove-based Input ». In : *IEEE Comput. Graph. Appl.* 14.1 (1994), p. 30–39 (cf. p. 132).

[208] Atsushi Sugiura et Yoshiyuki Koseki. « A User Interface Using Fingerprint Recognition: Holding Commands and Data Objects on Fingers ». In : *UIST '98*. New York, NY, USA : ACM, 1998, p. 71–79 (cf. p. 131, 133).

[209] Larry Tesler. « The Smalltalk Environment ». In : *Byte* (1981) (cf. p. 49–51, 58, 59, 65).

[210] H Thimbleby. « Modes, WYSIWYG and the von Neumann bottleneck ». In : *IEE Colloquium on Formal Methods and Human-Computer Interaction*. Fév. 1988, p. 4/1–4/5 (cf. p. 45, 48, 51, 53).

[211] Harold Thimbleby. « Character level ambiguity: consequences for user interface design ». In : *International Journal of Man-Machine Studies* 16.2 (fév. 1982), p. 211–225 (cf. p. 45, 48, 51–53).

[212] Touch Screen Middle East. « About Touch Screens » (cf. p. 14).

[213] Theophanis Tsandilas, Caroline Appert, Anastasia Bezerianos et David Bonnet. « Coordination of Tilt and Touch in One- and Two-handed Use ». In : *CHI '14*. New York, NY, USA : ACM, 2014, p. 2001–2004 (cf. p. 63).

[214] Theophanis Tsandilas et Wendy Mackay. « Knotty gestures: subtle traces to support interactive use of paper ». In : *Proceedings of the International Conference on Advanced Visual Interfaces*. New York, NY, USA : ACM, 2010, p. 147–154 (cf. p. 61, 80).

[215] Huawei Tu, Xiangshi Ren et Shumin Zhai. « A Comparative Evaluation of Finger and Pen Stroke Gestures ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 1287–1296 (cf. p. 60).

[216] Huawei Tu, Xing-Dong Yang, Feng Wang, Feng Tian et Xiangshi Ren. « Mode switching techniques through pen and device profiles ». In : *APCHI '12*. 2012, p. 169–176 (cf. p. 60, 63, 65, 87).

[217] D Umbach et K N Jones. « A Few Methods for Fitting Circles to Data ». In : *IEEE Transactions on Instrumentation and Measurement* 52.6 (déc. 2003), p. 1881–1885 (cf. p. 83, 84).

[218] P Viviani et T Flash. « Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. » In : *Journal of experimental psychology. Human perception and performance* 21.1 (1995), p. 32–53 (cf. p. 31).

[219] P Viviani et C Terzuolo. « Trajectory determines movement dynamics ». In : *Neuroscience* 7.2 (1982), p. 431–437 (cf. p. 31).

[220] Daniel Vogel et Patrick Baudisch. « Shift: A Technique for Operating Pen-based Interfaces Using Touch ». In : *CHI '07*. New York, NY, USA : ACM, 2007, p. 657–666 (cf. p. 21, 22, 58, 91, 130).

[221] Daniel Vogel et Géry Casiez. « Hand Occlusion on a Multi-touch Tabletop ». In : *CHI '12*. New York, NY, USA : ACM, 2012, p. 2307–2316 (cf. p. 21).

[222] Julie Wagner, Stéphane Huot et Wendy Mackay. « BiTouch and BiPad: designing bimanual interaction for hand-held tablets ». In : *CHI '12*. New York, New York, USA : ACM Press, mai 2012, p. 2317 (cf. p. 52, 53, 58, 125).

[223] Julie Wagner, Eric Lecolinet et Ted Selker. « Multi-finger Chords for Hand-held Tablets : Recognizable and Memorable ». In : *CHI '14*. New York : ACM, 2014, p. 2883–2892 (cf. p. 129–132).

[224] Robert Walter, Gilles Bailly et Jörg Müller. « StrikeAPose: Revealing Mid-Air Gestures on Public Displays ». In : *CHI '13*. New York, New York, USA : ACM, avr. 2013, p. 841–850 (cf. p. 55, 64, 81).

[225] Feng Wang, Xiang Cao, Xiangshi Ren et Pourang Irani. « Detecting and leveraging finger orientation for interaction with direct-touch surfaces ». In : *UIST '09*. New York, New York, USA : ACM Press, oct. 2009, p. 23 (cf. p. 18, 131).

[226] Jingtao Wang et John Canny. « FingerSense: Augmenting Expressiveness to Physical Pushing Button by Fingertip Identification ». In : *CHI '04 Extended Abstracts*. New York, NY, USA : ACM, 2004, p. 1267–1270 (cf. p. 129–131, 133).

[227] Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D Hollan et Jan Borchers. « SLAP widgets: bridging the gap between virtual and physical controls on tabletops ». In : *CHI '09*. New York, NY, USA : ACM, 2009, p. 481–490 (cf. p. 18).

[228] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo et Sarah Williams. « Rock & Rails: Extending Multi-touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations ». In : *CHI '11*. New York, NY, USA : ACM, 2011, p. 1581–1590 (cf. p. 18).

[229] Daniel Wigdor et Dennis Wixon. « Brave NUI World: Designing Natural User Interfaces for Touch and Gesture ». In : (avr. 2011) (cf. p. 32, 55, 81).

[230] Andrew D Wilson. « PlayAnywhere: A Compact Interactive Tabletop Projection-vision System ». In : *UIST '05*. New York, NY, USA : ACM, 2005, p. 83–92 (cf. p. 17).

[231] Andrew D. Wilson. « Robust computer vision-based detection of pinching for one and two-handed gesture input ». In : *UIST '06*. New York, New York, USA : ACM, oct. 2006, p. 255–258 (cf. p. 64, 65).

[232] Graham Wilson, Craig Stewart et Stephen A Brewster. « Pressure-based Menu Selection for Mobile Devices ». In : *MobileHCI '10*. New York, NY, USA : ACM, 2010, p. 181–190 (cf. p. 18).

[233] Jacob O. Wobbrock, Meredith Ringel Morris et Andrew D. Wilson. « User-defined gestures for surface computing ». In : *CHI '09*. New York, New York, USA : ACM Press, avr. 2009, p. 1083 (cf. p. 27–32, 116).

[234] Catherine G Wolf. « Can people use gesture commands? » In : *SIGCHI Bull.* 18.2 (oct. 1986), p. 73–74 (cf. p. 31, 32).

[235] Catherine G Wolf et Palmer Morrel-Samuels. « The Use of Hand-drawn Gestures for Text Editing ». In : *Int. J. Man-Mach. Stud.* 27.1 (juil. 1987), p. 91–102 (cf. p. 20, 27, 31, 32).

[236] Catherine G Wolf et James R Rhyne. « Gesturing with Shared Drawing Tools ». In : *INTERACT '93 and CHI '93 Conference Companion on Human Factors in Computing Systems*. New York, NY, USA : ACM, 1993, p. 137–138 (cf. p. 27).

[237] David D Woods. « The Price of Flexibility ». In : *IUI '93*. New York, NY, USA : ACM, 1993, p. 19–25 (cf. p. 51, 53).

[238] Mike Wu et Ravin Balakrishnan. « Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays ». In : *UIST '03*. New York, New York, USA : ACM Press, nov. 2003, p. 193–202 (cf. p. 18, 56, 64).

[239] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines et Ravin Balakrishnan. « Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces ». In : *Tabletop '06*. IEEE, jan. 2006, p. 185–192 (cf. p. 27, 55, 81).

[240] Robert Zeleznik et Timothy Miller. « Fluid inking: augmenting the medium of free-form inking with gestures ». In : *GI '06*. Canadian Information Processing Society, juin 2006, p. 155–162 (cf. p. 27, 57, 60–62).

[241] Shumin Zhai et Per Ola Kristensson. « Shorthand writing on stylus keyboard ». In : *CHI '03*. New York, New York, USA : ACM Press, avr. 2003, p. 97–104 (cf. p. 30, 33).

[242] Shumin Zhai, Per Ola Kristensson, Caroline Appert, Tue Haste Andersen et Xiang Cao. « Foundational Issues in Touch-Screen Stroke Gesture Design - An Integrative Review ». In : *Foundations and Trends in Human-Computer Interaction* 5.2 (2012), p. 97–205 (cf. p. 27–29, 31, 37, 38, 121, 123).

[243] Shengdong Zhao, Maneesh Agrawala et Ken Hinckley. « Zone and Polygon Menus: Using Relative Position to Increase the Breadth of Multi-Stroke Marking Menus ». In : *CHI '06*. New York, New York, USA : ACM Press, avr. 2006, p. 1077 (cf. p. 35, 36, 58).

[244] Shengdong Zhao et Ravin Balakrishnan. « Simple vs. Compound Mark Hierarchical Marking Menus ». In : *UIST '04*. New York, NY, USA : ACM, 2004, p. 33–42 (cf. p. 35, 36, 57).

Techniques d'interaction pour applications riches sur tablettes multi tactiles

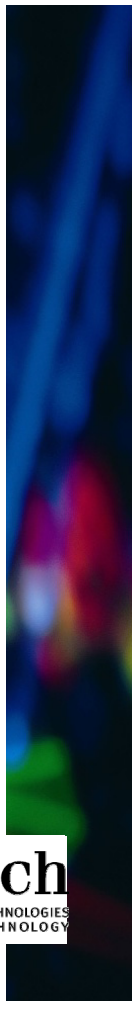
Quentin Roy

RESUME : L'objectif de cette thèse CIFRE en collaboration avec *GE HealthCare* est de contribuer à enrichir l'interaction sur tablettes tactiles en rendant utilisables sur ces surfaces réduites un plus grand nombre de fonctionnalités. Dans le premier volet de cette thèse nous nous sommes efforcés d'optimiser l'exploitation des canaux d'entrées existants sur tablette en concevant et en évaluant expérimentalement de nouvelles techniques d'interaction : le *Sigma-Menu*, le *Finger-Menu*, les *Augmented Letters* et la *Multi-Touch Menubar*. Dans le second volet de cette recherche, nous avons évalué la valeur ajoutée de l'identification des doigts à l'interaction mono-contact avec pour objectif d'étendre les canaux d'entrée des tablettes tactiles.

MOTS-CLEFS : multitouch, tablette tactile, mode, délimiteur, identification des doigts, doigt

ABSTRACT : This PhD in collaboration with *GE HealthCare* aims at improving interaction on multitouch tablets so as to allow these devices to support applications with a larger number of functionalities. The first part of this research aimed at optimizing the use of extant input channels of multitouch tablets by designing and experimentally evaluating a number of new interaction techniques : the *Augmented Letters*, the *Sigma-Menu*, the *Finger-Menu* and the *MultiTouch Menubar*. The second part of this research focused on the evaluation of the added value of finger identification to single touch interaction as a way to extend the current input channels of multitouch tablets.

KEY-WORDS : multitouch, tablet, mode, delimiter, finger identification, finger



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY