HAL
open science

Techniques d'assistance à la saisie de texte sur périphériques mobiles dans le cas de la déficience visuelle

Philippe Roussille

To cite this version:

Philippe Roussille. Techniques d'assistance à la saisie de texte sur périphériques mobiles dans le cas de la déficience visuelle. Informatique mobile. Université Paul Sabatier - Toulouse III, 2017. Français. NNT : 2017TOU30020 . tel-01743690

HAL Id: tel-01743690

https://theses.hal.science/tel-01743690v1

Submitted on 26 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *27/01/2017* **par :**

Philippe Roussille

Techniques d'assistance à la saisie de texte sur périphériques mobiles dans le cas de la déficience visuelle

---

JURY

Edwige Pissaloux	Professeur - Univ. de Rouen	Rapporteur
Benoît Martin	Professeur - Univ. Lorraine	Rapporteur
Anke Brock	CR - INRIA Bordeaux	Examinateur
Emmanuel Dubois	Professeur - Univ. Toulouse	Examinateur
Mathieu Raynal	MCF - Univ. Toulouse	Directeur de thèse

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur de Thèse :
Mathieu Raynal
Rapporteurs :
Benoît Martin et *Edwige* Pissaloux

Remerciements

Un travail ne se fait jamais seul, aussi je crains de manquer de place pour pouvoir rendre compte de ce que j'ai reçu à ceux qui m'ont aidé pour la réalisation de ce manuscrit.

Dans un premier temps, je souhaiterais remercier les membres du jury. Merci à mes examinateurs, Emmanuel Dubois et Anke Brock, pour leur présence au sein du jury. Merci à mes deux rapporteurs, Benoît Martin et Edwige Pissaloux, d'avoir pris le temps et l'énergie nécessaire à la correction du manuscrit.

Je voudrais remercier tout particulièrement Mathieu, mon directeur de thèse pour son soutien ô combien précieux au cours de ce travail Merci de m'avoir permis de commencer dés 2012, alors que rien n'était encore certain ; de m'avoir donné la chance de pouvoir débuter, réaliser et achever ce travail de recherche. Merci de m'avoir toujours accompagné, même et surtout dans les moments de découragement, de m'avoir poussé à toujours aller plus loin au-delà des limites que je pensais initialement atteignables. Merci pour tes longues soirées passées à la correction et à la rédaction (autant pour les articles que pour ce document), aux nombreuses aides à la réalisation et la relecture du manuscrit, à ton accompagnement sans faille et ton implication constante à mes cotés. Merci !

Je remercie également mes relecteurs : Gary, Bernard, Mathieu ; qui ont contribué à la finalisation du manuscrit. Merci aussi à l'ensemble des membres de l'équipe Elipse, qui m'ont supporté (dans tous les sens du terme !) avec un même dynamisme pendant ces trois ans. Merci à Emmanuel, Marcos, Mustapha, Philippe, Marc, Antonio, Bernard, Christophe, Nadine, Frédéric, Philippe et Mathieu. Une pensée toute particulière aux doctorants, passés, présents et futurs, que j'ai eu l'occasion de cô-

toyer durant toute ma thèse, notamment Anke, Slim, Laurent, Louis-Pierre, Damien, Adrien, Grégoire, Olivier, Damien, Lilian, Victor ; ainsi qu'à Gary, Sandra, Mathias, et Damien. Enfin, une pensée toute particulière à Matthieu, Julie et Houssem, qui partagent avec moi le bureau 270 et qui supportent avec fort stoïcisme mon humour. Bon courage à tous, les soutenances approchent !

Merci à Claude Griet et l'ensemble des personnes de l'Institut des Jeunes Aveugles pour leur participation à mes expériences. Merci à Thierry Couspeyre et l'ensemble des membres de l'institut Lestrade pour leur contribution à l'étude longue réalisée sur deux semaines.

Un merci également à toute l'équipe accompagnante de l'IRIT, de l'EDMITT, et de l'Université : merci à Floriane et à Françoise, les sourires qui accueillent toujours avec plaisir ; merci à Thierry, Guillaume, Michael, Christophe, Jean-François, Jacques, William, Martine, Brigitte, Anne-Marie, Romain, Annie et tous les autres, pour leurs longues discussions d'un Linuxien dans la recherche en pays Windowsien. Merci à Christine, pour m'avoir donné la première occasion d'enseigner. Merci à Mathieu pour m'avoir fait confiance pour ses cours et permis de passer de R à Python. Merci au docteur Laurence Cadieux et à son équipe pour l'attention qu'elles apportent au niveau des doctorants en situation de handicap. Merci à Agnès et à Martine pour leur suivi au sein de l'école doctorale. Merci à Arièle, Isabelle, Maryline, Pascale et Michèle et tout le service de l'UPS pour leur patience.

Une pensée toute particulière et très précieuse à Benoît et son équipe de l'Upsidium, avec Philippe, Cindy, Eddie, Laurent, Patrick, Joseph, Christine, Elly, Geneviève, Cathy, Tahar, Laurie, Rosalinda et Didier, sans qui je n'aurais jamais pu me sustenter, à midi ou en soirée. Merci à eux pour leur enthousiasme constant, leur serviabilité et leur bonne humeur communicative.

Merci à mes étudiants, de m'avoir permis de me remettre en question lorsque c'était nécessaire. Merci à mes collègues de cours pour leur soutien et les longs échanges pédagogiques.

Merci à M. Ayache et à M. Daydé qui m'ont orienté vers l'IRIT, où j'ai pu faire ma thèse. Merci à M. Miles et ses collaborateurs qui m'ont offert ma chance à Lyon. Merci à M. Roussel pour son accompagnement pendant toutes ces années de thèse.

Merci à M. Gourinat pour ses encouragements.

Merci à Médéric, pour m'avoir initié un peu plus à mon penseur favori. Merci au groupe scout des dominicains, pour avoir égayé mes week-ends et m'avoir permis de me rendre compte qu'on peut tous faire grandir et que, peu importe la situation, chacun peut apporter quelque chose.

Merci à Gary, de m'avoir suivi jusqu'au bout dans mon aventure rôliste sous le fanion de l'UPSJDR, et à ceux qui tiennent le cap sans nous, maintenant. Merci à Arnaud pour ses longues discussions. Merci au tissu associatif de l'Université Paul Sabatier pour leurs encouragements.

Je souhaite mentionner avec beaucoup d'affection M. Jack Sagot, grâce à qui l'expertise et le soutien offerts, depuis des années, aux jeunes en situation de handicap m'a permis de me donner l'ambition et la motivation nécessaires pour aller toujours de l'avant.

J'ai enfin une pensée toute particulière pour M. Cédric Villani, rencontré il y a quatre ans, alors que j'étais encore en attente de l'obtention de ma bourse : nos échanges, et sa protection bienveillante m'ont permis d'apprendre énormément sur le monde de la recherche. Merci à lui pour la motivation qu'il a su me donner. Merci à la fondation de la Banque Populaire de m'avoir permis cette rencontre.

On commence un long travail par de bonnes fondations, et c'est sur ces dernières que reposent tout l'édifice : aussi, je me permets de dire un merci tout particulier à Christelle et à Laure, ainsi qu'à chacune de mes assistantes de vie. Merci à mes professeurs de lycée, sans qui je n'aurais jamais pu connaître ce que voulait dire « être passionné » par la matière que l'on enseigne. Merci à mes professeurs de prépa, sans qui je n'aurais jamais pu me rendre compte qu'on ne baisse pas les bras devant le travail, mais seulement une fois ce dernier réalisé. Merci à mes professeurs d'école d'ingénieur pour m'avoir permis de découvrir la passion de la recherche par le biais de vos enseignements. Merci à l'équipe du LIA de l'EPFL pour m'avoir fait goûter aux premières joies de la recherche et de la publication.

Un travail n'est jamais fait seul, et n'est jamais fait sans embûches. Ce n'est souvent que dans l'adversité qu'on découvre son véritable potentiel : je dédie une partie de ce travail à ceux qui ne croient pas assez en eux-mêmes, pour qui les études

supérieures sont un calvaire ou une impossibilité Merci à ceux qui m'ont mis des bâtons dans les roues et ont essayé de me décourager : on ne se rend compte de ce que l'on peut espérer uniquement lorsqu'on essaie de vous prendre votre futur ! Merci à mes amis d'enfance et à ceux qui m'ont permis de me construire, dès le lycée, et qui me soutiennent encore dans leur action. Merci à ma famille pour son soutien. Merci à ceux que j'ai pu croiser, et qui d'un sourire ou d'un geste sympathique m'ont motivé à aller toujours plus de l'avant. Je ne vous oublie pas.

Une petite pensée à toi que je ne connais pas encore, et qui lira ce manuscrit.

Enfin, il y a des personnes adorables que je ne peux pas oublier, car sans eux ce travail n'aurait jamais vu le jour... les plus importantes à mes yeux. Je remercie tout particulièrement ma maman, Roselyne, et mon papa, Jean-Michel, pour leur soutien constant. Sans vous je ne serai pas, sans vous je ne suis rien. Merci de m'avoir permis de vivre et d'exister, de m'avoir toujours suivi dans toutes les situations, les meilleures comme les pires, en me transmettant, avec vos encouragements, cette volonté qui m'anime. Merci de m'encourager dans les moments les plus difficiles ! À vous, je vous dédie ce manuscrit. Je vous aime très fort.

Table des matières

Introduction 15

I Saisie de texte sur dispositif mobile 21
1 Méthodologie d'évaluation 21
1.1 Critères d'évaluation 22
1.2 Mesures 25
2 Systèmes de saisie de texte sur dispositifs mobiles 39
2.1 Systèmes de reconnaissance de parole 40
2.2 Systèmes de reconnaissance de forme 40
2.3 Claviers logiciels 42
2.4 Claviers adaptés pour une saisie sans regard 51
2.5 Claviers Braille 53
3 Synthèse 57
3.1 Problème de la saisie Braille 57
3.2 Problème du retour visuel et de « *l'exploration douloureuse* » 58

II DUCK : un clavier déductif 61
1 Fonctionnement d'un clavier logiciel oralisé 62
1.1 Phase de saisie 62
1.2 Gestion des erreurs 64
2 Clavier DUCK 67
2.1 Choix de conception 67
2.2 Choix des interactions 69
2.3 Fonctionnement 70
2.4 Implémentation 71
2.5 Hypothèses 73

III Première étude expérimentale 75
1 Protocole 75
1.1 Participants 75
1.2 Dispositif 76
1.3 Procédure 77
1.4 Conception 77
1.5 Données collectées 78
2 Résultats 78
2.1 Vitesse de frappe pour les mots et les phrases 78
2.2 Décomposition de la saisie lors de l'utilisation de DUCK . . . 80
2.3 Position du mot dans la liste 82
2.4 Niveau de correction de la saisie de texte 83
2.5 Préférences des utilisateurs 84
3 Discussions 85

IV Interaction avec la liste de prédiction 87
1 Présentation des éléments 89
1.1 Dispositions proposées 89
1.2 Protocole 93
1.3 Résultats 95
1.4 Discussion 100
2 Retour audio 102
2.1 Retours audio étudiés 104
2.2 Protocole 105
2.3 Résultats 106
2.4 Discussion 110
3 Technique de validation 110
3.1 Protocole 111
3.2 Résultats 112
3.3 Discussion 114
4 Synthèse 115

V La gestion des mots courts . . . 117
1 Protocole . . . 121
1.1 Hypothèses . . . 121
1.2 Participants . . . 121
1.3 Dispositif . . . 121
1.4 Procédure . . . 122
1.5 Conception . . . 122
1.6 Données collectées . . . 122
2 Résultats . . . 123
2.1 Temps . . . 123
2.2 Taux d'erreurs . . . 124
2.3 Retour utilisateur . . . 125
3 Discussion . . . 126

VI Découverte et apprentissage de DUCK pour un usage journalier . . . 129
1 Améliorations apportées au système DUCK . . . 130
1.1 Intégration du mode mots courts . . . 130
1.2 Saisie glissée . . . 130
1.3 Interaction avec les listes . . . 131
2 Protocole . . . 131
2.1 Hypothèses . . . 131
2.2 Participants . . . 131
2.3 Matériel . . . 132
2.4 Procédure . . . 132
2.5 Conception . . . 133
2.6 Données collectées . . . 134
3 Résultats . . . 134
3.1 Vitesse de saisie de texte . . . 134
3.2 Précision des claviers . . . 139
3.3 Avis des utilisateurs . . . 142
4 Discussion . . . 144

Conclusion 147

Ouvertures 149

Appendices 153

A Consignes données aux participants 155
1 Première étude sur le clavier DUCK 155
1.1 Lors de l'accueil des participants 155
1.2 Lors de la phase de familiarisation 155
1.3 Lors de la phase d'évaluation 156
2 Étude sur l'interaction dans les listes 157
2.1 Lors de l'accueil des participants 157
2.2 Lors de la phase de familiarisation 157
2.3 Lors de la phase d'évaluation 157
3 Étude sur les mots courts 157
3.1 Lors de l'accueil des participants 157
3.2 Lors de la phase de familiarisation 158
4 Étude longue 158
4.1 Lors de l'accueil des participants 158
4.2 Lors de la phase de familiarisation 159
4.3 Lors de la phase d'évaluation 159

B Première tentative de réduction du problème d'exploration par l'utilisation d'arbres 161

C Génération de corpus 163
1 Outil : `corpus_study.py` 164
2 Corpus proposé 165

D Liste des mots courts 173

Table des figures

1 Évolution du téléphone portable vers le smartphone au cours des 25 dernières années 16

2 En haut, l'alphabet Graffiti ; en bas, Unistroke. 41

3 Le système No-Look-Notes : sélection du groupe de lettres (1,2) et choix de la lettre B (3,4). 45

4 Le clavier QuickWriting : à gauche, saisie de f ; à droite, saisie de "the". 47

5 Bimanual Gesture Keyboard : saisie de « *life* » sur le clavier. 48

6 Le système ROTEXT : l'inclinaison du téléphone associée à une secousse permet de choisir un caractère (ici, E, T ou S). 49

7 Le système TiltType : l'appui sur les quatre boutons extérieurs combiné à l'inclinaison du dispositif permet la saisie du caractère souhaité. 50

8 Le clavier BrailleType. 54

9 Le clavier BrailleTouch. 54

10 L'approche TypeInBraille. 55

11 La technique IFD : la main gauche correspond aux points Braille 1,2,3 tandis que la main droite accède aux points 4,5,6. 57

12 Utilisation de la correction automatique 65

13 Utilisation de listes de prédiction 66

14 Vitesse de frappe moyenne (en cps) en fonction de la longueur de mots pour les systèmes de saisie DUCK (en rouge) et VODKA (en bleu) . 80

15 Temps moyens (en s) en fonction des longueurs de mots. Les temps de frappe (T_{TAP}) et de validation (T_{VAL}) sont également indiqués ($T_D = T_{TAP} + T_{VAL}$) 81

16 Disposition absolue : agencement en A) ligne, B) colonne, et C) grille 90
17 Disposition relative 91
18 Exemple d'une sélection relative à 6 éléments 92
19 Disposition par page 93
20 Nombre d'éléments survolés 96
21 Temps moyen de sélection en secondes 97
22 Temps moyen de sélection en secondes en fonction de la position pour 4 éléments (à gauche) et pour 6 éléments (à droite) 98
23 Taux d'erreurs à la sélection 99
24 Scores SUS 100
25 Éléments parcourus 107
26 Temps de sélection 108
27 Erreur par technique 109
28 Temps par geste 113
29 Erreur par geste 114

30 Le mode de validation à frappe double 119
31 Le mode de validation à deux doigts 120
32 La saisie glisser-taper 120
33 Temps mis pour taper un mot, en fonction de la technique et de la longueur du mot 124
34 Taux d'erreurs à la saisie d'un mot en fonction de la technique et de la longueur du mot 125

35 Vitesse de saisie des deux claviers à l'issue de la première et de la deuxième session 135
36 Vitesse de saisie des deux claviers à l'issue de la première session . . . 136
37 Vitesse de saisie des deux claviers à l'issue de la deuxième session . . 137

38 Évolution du temps de pause intercaractère 138
39 Réussite par clavier 140
40 Utilisation de la fonction mots courts 142
41 Évolution des scores SUS pour les deux claviers 143

Sicut enim maius est illuminare quam lucere solum, ita maius est contemplata aliis tradere quam solum contemplari.

Il est plus beau d'éclairer que de briller seulement ; de même est-il plus beau de transmettre aux autres ce qu'on a contemplé que de contempler seulement.

Thomas d'Aquin

Introduction

Aujourd'hui, les ordinateurs sont monnaie courante. De plus en plus petits et portatifs, l'évolution est impressionnante : des gros ordinateurs des salles prenant 162 m^2, nous sommes arrivés pour la plupart des ordinateurs modernes à des appareils portables. En sus de la réduction de leur taille, ils sont devenus de plus en plus puissants, et de plus en plus abordables. Alors qu'il était impensable à l'époque de pouvoir réaliser des actions plus complexes que de la saisie de texte et du calcul simple, nous avons aujourd'hui des dispositifs capables de véritables prouesses techniques avec une explosion de possibilités, de l'envoi de SMS ou d'emails à une interaction en 3D[Marek et al., 2016], pour atteindre de véritables challenges, allant jusqu'à imiter et reproduire les comportements humains[Amblard and Boumaza, 2014], simuler l'équivalent de 13 milliards d'années d'évolution cosmique[Vogelsberger et al., 2014], tout en utilisant une mémoire multiferroïque qui n'est plus magnétique[Heron et al., 2014], tandis que la quantité d'information échangée entre les réseaux devient colossale[Young, 2014].

Les techniques d'interaction de ces dispositifs ont également évolué. Initialement, seules les cartes perforées[Norberg, 1990] servaient à la fois de périphérique d'entrée-sortie et de stockage de masse. Le système d'affichage était très rudimentaire : du simple tube cathodique à rayon de 1922, l'écran s'est peu à peu démocratisé sous le moniteur cathodique (1954). Rapidement, les écrans LCD sont élaborés dès 1986, et les années 2000 ont marqué leur apparition auprès du grand public. En parallèle de la miniaturisation des dispositifs, les premiers écrans tactiles à matrice organique amorcent la transition vers les téléphones mobiles dès le début des années 2000.

L'évolution des téléphones mobiles s'est faite en deux phases (voir Figure 1). Dans

un premier temps, ces dispositifs ont progressivement diminué leur taille en utilisant des claviers physiques de plus en plus réduits et de petits écrans. Puis l'apparition des écrans tactiles a peu à peu remplacé les claviers physiques et la taille des dispositifs a de nouveau augmenté pour proposer des écrans de plus en plus grands et aboutir à l'émergence des tablettes tactiles. Le vieux téléphone mobile à touches a rapidement été délaissé au profit de ces dispositifs munis d'écrans tactiles de plus en plus grands. Les touches physiques statiques sont alors remplacées par des touches virtuelles dans des interfaces où la majorité des interactions s'effectuent directement sur l'écran avec le doigt ou un stylet.

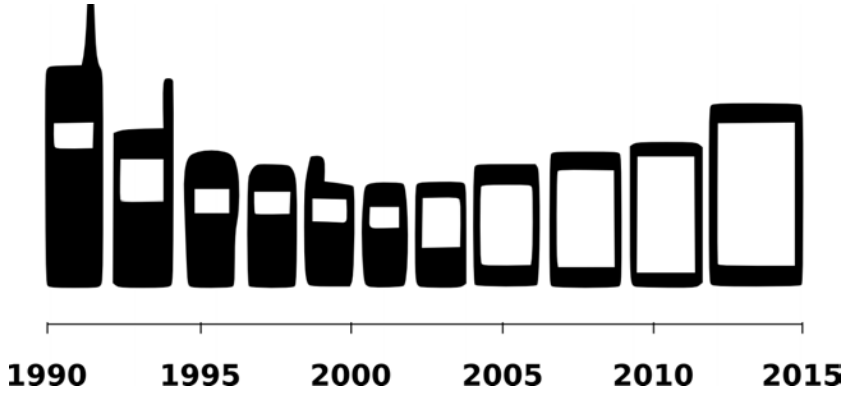

FIGURE 1 – Évolution du téléphone portable vers le smartphone au cours des 25 dernières années

La transformation vers le tout tactile sur de tels dispositifs s'est accompagnée de nouvelles habitudes de communication : initialement réservés à la téléphonie et la saisie de courts messages textuels, pour les téléphones portables d'un côté ; et à la prise de rendez-vous pour les PDAs (Personal Digital Assistants, assistants personnels digitaux) de l'autre, l'apparition de la communication sans fil et la démocratisation de l'Internet a permis de combiner leurs fonctions en un nouveau type de dispositif, les « *smartphones* » (téléphones intelligents). Ces dispositifs ont rendu la communication écrite de plus en plus habituelle et incontournable, que cela soit par la saisie de rendez-vous, la navigation sur internet, la communication sur les réseaux sociaux, ou l'envoi de courriers électroniques. Mais les smartphones ne sont pas les seuls dispo-

sitifs en vogue : les tablettes tactiles, vues comme l'intersection entre le smartphone et l'ordinateur portable, très légères et portables, permettent une utilisation bureautique en situation de mobilité. Il devient ainsi possible de saisir des rapports dans une salle d'attente ou dans un avion, par exemple, sans encombrement. Enfin, alors que la montre semblait disparaître peu à peu, un nouveau dispositif est apparu pour la remplacer : la montre connectée. De la même manière que le smartphone et la tablette, elle utilise un écran tactile de petite taille, directement utilisable sur un poignet. Il est ainsi possible d'entrer des informations ou de réaliser des actions basiques, comme consulter un service météorologique en ligne, lire et répondre à un message textuel ou un courrier électronique, de faire du suivi de santé ou sportif, mais aussi de consulter ses rendez-vous.

Bien que l'écran tactile permette un bon retour de l'information qui y est présentée, de tels dispositifs forcent alors un changement de paradigme : sur un clavier ou un téléphone à touches, le moyen de saisir du texte est simple d'accès. Les touches sont statiques, facilement localisables et par conséquent mémorisables. Leur relief et leur taille permettent également de facilement y accéder. Sur un petit dispositif entièrement tactile, il est possible d'imiter un clavier physique, en utilisant un clavier dit logiciel, reprenant un fonctionnement similaire au clavier physique et simulant des touches que l'utilisateur peut activer au moyen d'un stylet ou d'un doigt. Cependant, si pour les voyants les animations et retours visuels proposés sur ces claviers, les rendent facilement utilisables, le manque de relief pénalise fortement l'utilisation de ce type de clavier dans un mode d'utilisation « sans regard ». D'autre part, ces claviers ont de petites touches, ce qui pénalise la précision de la saisie de texte. En revanche, ces claviers étant logiciels, il est possible de les coupler facilement avec des systèmes évoluant dynamiquement en fonction de la saisie de l'utilisateur tels que les systèmes de prédiction ou de déduction. Ceux-ci permettent alors d'améliorer les performances de saisie de texte de l'utilisateur.

Ces dispositifs se veulent facilement utilisables pour le grand public mais, les interactions imposées par un écran tactile ne sont pas universelles : une partie de la population est pénalisée par ce type de dispositif. Du fait de la petite taille de l'interface et des problèmes de précision que cela engendre, les personnes en situation de

déficience motrice peuvent difficilement pallier efficacement ces obstacles supplémentaires que ces dispositifs impliquent. Dans le cas de la déficience visuelle, le problème est différent : le dispositif, du fait de la présence d'un écran tactile, est entièrement lisse. Il ne présente aucune aspérité ou particularité permettant aux non-voyants de distinguer une zone particulière d'une autre sur la zone principale d'interaction. Ils ont alors très peu de repères tactiles : toutes les interactions sont spatialement associées à des éléments affichés à l'écran. Les déficients visuels sont donc contraints d'explorer, avec leur doigt, l'intégralité de l'environnement afin de trouver les éléments avec lesquels ils veulent interagir.

Dans le cas de la saisie de texte sur ces dispositifs entièrement tactiles, le problème est amplifié. Les claviers logiciels sont un ensemble de touches numériques. Dans ce contexte, l'interaction la plus utilisée par les déficients visuels pour pouvoir saisir du texte avec de tels dispositifs est basée sur l'exploration de l'écran avec le doigt couplée à une oralisation par synthèse vocale du caractère situé sous le doigt [1]. Par conséquent, les utilisateurs malvoyants doivent constamment explorer l'écran pour atteindre une touche spécifique. Bien que le retour vocal puisse être amélioré avec des indices vibratoires, ce processus, appelé « exploration douloureuse » [Bonner et al., 2010], est lent et nécessite beaucoup d'efforts. D'autres solutions ont également été conçues pour répondre plus spécifiquement aux besoins des non-voyants. Cependant, ces solutions, par exemple basées sur l'alphabet Braille, nécessitent généralement un temps d'apprentissage non négligeable qui freine les utilisateurs à abandonner l'usage d'un clavier logiciel standard dont ils connaissent parfaitement la disposition des caractères, du fait de leur utilisation des claviers physiques sur ordinateur.

La saisie de texte étant un élément prépondérant et essentiel pour pouvoir communiquer au moyen de ces dispositifs mobiles, nous avons focalisé nos travaux de recherche sur l'optimisation de la saisie sur dispositifs tactiles pour les déficients visuels. Nous allons ici présenter l'approche que nous avons choisie tout au long de ce projet de thèse pour répondre aux problématiques de saisie de texte sur ces dispositifs. Nous choisissons de nous focaliser et de nous limiter seulement au périphérique

1. Les systèmes de saisie les plus couramment utilisés avec ce type d'approche sont VoiceOver d'Apple Inc., ou TalkBack de Google Inc.

mobile en lui-même, sans modification ou augmentation de ce dernier par des périphériques annexes (par exemple, un clavier Bluetooth), ni sans adaptation partielle ou complète (par exemple, un cache écran ou un guide doigt).

Le chapitre 1 dresse une analyse des différents types de systèmes de saisie de textes existants sur périphériques mobiles, ainsi que les mesures nécessaires à leurs études. Nous avons étudié les systèmes offrant une approche non visuelle, mais aussi ceux spécifiquement destinés à une utilisation par des déficients visuels. De cette analyse, nous avons dressé une liste des principaux problèmes qui nous semblaient essentiels d'étudier afin d'améliorer la saisie par des utilisateurs non-voyants avec un clavier logiciel standard.

Le problème d'« *exploration douloureuse* » (*painful exploration*) que nous avons décrit précédemment ressort comme le problème majeur d'interaction avec un clavier logiciel et a constitué le principal axe de recherche de nos travaux. Cette exploration du clavier est longue, fastidieuse et coûteuse pour l'utilisateur, qui doit la reproduire à chaque entrée d'un nouveau caractère, sans compter les corrections et fautes de frappe. Le chapitre 2 présente la solution que nous proposons pour répondre à ce problème. Il s'agit du système DUCK basé sur une saisie approximative de l'utilisateur, qui est corrigée au fur et à mesure des mots saisis par un système de déduction.

Après avoir conçu et développé un premier prototype de notre système, nous avons réalisé une première évaluation afin de comparer les performances des utilisateurs avec notre système par rapport aux performances obtenues avec un clavier utilisant l'exploration douloureuse. La présentation de cette étude et l'analyse de ses principaux résultats constituera notre troisième chapitre.

De cette évaluation, nous avons dégagé deux problèmes qu'il nous paraissait important d'étudier pour améliorer les performances de notre système.

Le premier problème concerne les différents moyens d'interaction mis en œuvre pour faciliter la sélection non visuelle d'un mot dans une liste de mots. En effet, que ce soit pour notre système DUCK, ou pour la plupart des claviers logiciels sur mobile, les listes sont très fréquemment utilisées pour améliorer les performances de saisie : par exemple, une liste peut fournir une possible complétion du mot (liste de prédiction), permettre de rectifier l'orthographe d'un mot (liste de correction), ou

anticiper intégralement une future frappe en fonction des frappes précédentes (liste de déduction). L'utilisation non visuelle de ces listes pose des problèmes lors des phases d'exploration, de sélection et de validation des mots dans la liste. Nous avons mené trois études afin de proposer les interactions les plus adaptées pour ces trois phases. Le chapitre 4 présentera ces trois études et les principaux résultats obtenus.

Le second problème soulevé par l'utilisation du système DUCK est la saisie des mots courts, c'est à dire les mots de quatre caractères au plus. Ces mots, dont certains sont très fréquemment utilisés, nécessitent de pouvoir être saisis plus rapidement. Dans le chapitre 5, nous proposons un mécanisme spécifique permettant d'accéder plus rapidement aux mots courts les plus fréquemment utilisés dans la langue française.

Nous avons achevé notre travail en effectuant une dernière étude avec quatre utilisateurs non voyants afin de valider notre système intégrant les améliorations proposées lors des deux chapitres précédents. De plus, cette étude a été réalisée en deux sessions séparées de 15 jours afin d'étudier l'évolution des performances des utilisateurs avec une phase d'apprentissage. Le chapitre 6 présentera les principaux résultats de cette étude.

Enfin, nous conclurons ce mémoire par une synthèse des travaux effectués et nous proposerons plusieurs ouvertures possibles comme suite à donner à ces travaux sur la saisie de texte sur dispositif mobile dans le cas de la déficience visuelle.

Chapitre I

Saisie de texte sur dispositif mobile

Pour bien débuter notre étude, il est nécessaire de l'axer suivant deux points. En effet, comme nous comptons répondre à une problématique liée à la saisie de texte, nous souhaitons pouvoir analyser les différents types de systèmes de saisie de texte existants en étudiant comment ils permettent de répondre aux besoins des utilisateurs et quels sont les moyens de les comparer. Nous allons de ce fait commencer par une étude de la méthodologie couramment employée pour évaluer les systèmes de saisie de texte ainsi que les différents critères et mesures que cette dernière requiert.

Puis nous allons ensuite nous intéresser aux différents systèmes de saisie de texte existants sur dispositifs mobiles, et plus particulièrement à leurs utilisations dans le contexte de la saisie non visuelle et ceux utilisés par des déficients visuels, afin de pouvoir cartographier les solutions existantes et les différents problèmes qu'ils posent.

1 Méthodologie d'évaluation

Pour permettre une évaluation cohérente des différentes techniques de saisie de texte, il existe de nombreux critères visant à utiliser une méthodologie d'évaluation correcte et stricte permettant de réaliser des évaluations et des comparaisons entre les différents systèmes de saisie.

1.1 Critères d'évaluation

Il est nécessaire, dans le cadre d'une étude orientée vers l'interaction Homme-Machine (et donc, encore plus dans le cas d'études destinées aux utilisateurs en situation de déficience) de mesurer l'utilisabilité de la méthode de saisie choisie. Selon la norme ISO 9241-11 [International Organization for Standardization, 1998] (« *lignes directrices relatives à l'utilisabilité* »), on peut définir l'utilisabilité comme étant « *le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié* ». Cette définition met en jeu trois facteurs pour mesurer le degré d'utilisabilité :

1. *efficacité* : Le but est-il atteint ? Des critères de réussites doivent être définis.
2. *efficience* : Quels efforts sont nécessaires pour atteindre le but ? Plus les efforts sont faibles, plus l'efficience est grande.
3. *satisfaction* : Quel ressenti l'utilisateur a-t-il de son expérience ?

En reprenant l'analyse de Zhai et Kristensson[Zhai et al., 2005], les spécificités de ces critères dans le cas de la saisie de texte sont à rapporter aux processus cognitifs, perceptifs et moteurs complexes qui y sont propres. Du fait de cette complexité, il n'y a pas vraiment d'évaluation standard ou de référence en la matière. Même dans le cas de la saisie traditionnelle, il est difficile de trancher sur l'efficacité réelle d'un agencement de clavier par rapport à un autre[Cooper, 2012]. Il est donc difficile d'établir une liste objective de critères stricts qui permettraient de valider définitivement la méthode expérimentale. Néanmoins, il est cependant nécessaire, pour toute contribution majeure dans le domaine, de considérer *a minima* les aspects et les problèmes suivants.

Le premier aspect qui nous intéresse est celui de la performance ultime : c'est-à-dire la performance théorique maximale que peut espérer atteindre un utilisateur avec un système de saisie de texte. Cet objectif est souvent le plus important lors de la conception d'une technique de saisie de texte. La vitesse effective est souvent la première question qui vient à l'esprit d'un utilisateur qui rencontre une nouvelle méthode de saisie. Il est, comme nous le verrons, très difficile de répondre nettement

à cette question : la vitesse de frappe de l'utilisateur dépend de son taux de maîtrise du système, et de nombreuses autres variables telles que les erreurs de saisie ou l'attention nécessaire au cours de la saisie. Sans un certain temps d'apprentissage, ce qui est mesuré lors d'une expérience est rarement ce qui sera la performance "ultime".

Le deuxième aspect concerne la performance initiale et le temps d'apprentissage. Comme mentionné précédemment, la performance initiale d'un utilisateur dit novice et le début de sa courbe d'apprentissage sont extrêmement déterminants pour la suite de l'utilisation du système de saisie de texte par un utilisateur. En effet, la majeure partie d'une population d'utilisateurs ne souhaite pas investir un nombre d'heures conséquent pour pouvoir bénéficier des avantages d'une nouvelle interface. D'autre part, le temps nécessaire à l'apprentissage n'est jamais sans conséquence, et nécessite généralement beaucoup de temps pour pouvoir affirmer qu'un utilisateur est devenu expert d'un système et qu'il en a atteint son maximum en terme de performance. Par exemple, dans le cas des agencements[Cooper, 2012], il est nécessaire d'effectuer plusieurs heures d'apprentissage sur le nouvel agencement pour arriver à une vitesse comparable à l'ancien agencement. Une solution possible est de jalonner l'apprentissage en plusieurs étapes (15 minutes, 1 heure, 5 heures, etc.)[Kjeldskov and Stage, 2004]. Sinon, il est toujours possible de prédire la performance future d'un utilisateur avec un système en réalisant une étude basée sur une régression linéaire [Crossman, 1959], mais une telle approche nécessite une quantité suffisante d'essais [Zhai et al., 2002], et ne sera toujours qu'une estimation théorique de la performance que pourra atteindre un utilisateur.

Le troisième aspect concerne les erreurs en tant que telles et la gestion de ces erreurs. En effet, il est important de considérer le taux d'erreurs et la facilité à corriger l'erreur. Cet aspect aura aussi un rôle important sur la vitesse effective de l'utilisateur, c'est-à-dire sa vitesse de saisie en prenant en compte sa gestion des erreurs. Il existe plusieurs possibilités concernant la gestion des erreurs dans une étude expérimentale, qui impactent les résultats de l'étude dans tous les cas. Il est ainsi possible de laisser les erreurs dans le texte [MacKenzie and Zhang, 1999a], de les rendre impossibles [Accot and Zhai, 2002], ou de demander aux participants de les corriger[MacKenzie, 2002a]. Le temps mesuré en conséquence dépend alors du

mécanisme de correction mis en place. Il n'y a donc aucune méthode universelle permettant d'obtenir une équivalence directe erreur/vitesse.

L'**attention visuelle et cognitive** constitue le quatrième aspect. Idéalement, une méthode de saisie de texte ne demande que très peu (ou pas) d'attention, laissant à l'utilisateur la possibilité de se concentrer sur sa saisie, sur son environnement, ou sur d'autres tâches. La majorité des systèmes de saisie de texte sur dispositif tactile, du fait du manque de repères tactiles, nécessitent plus d'attention visuelle que les claviers physiques, où il est possible de "taper à l'aveugle" grâce au relief des différentes touches. Lorsqu'une méthode de saisie nécessite une attention visuelle, il est difficile de mesurer le coût de cette attention supplémentaire demandée à l'utilisateur.

Le cinquième et dernier aspect touche aux définitions subjectives par l'**appréciabilité**. Une méthode de saisie ne doit pas être uniquement efficace, mais également agréable à utiliser. Dans le cas de la saisie de texte, est-ce la fluidité de la méthode, la facilité d'utilisation, la facilité d'apprentissage, à l'image du défi lancé au joueur dans le cas de jeux vidéo, qui font l'attrait d'une technique ? Comment peut-on mesurer le retour utilisateur quant à l'« *attractivité* » d'une technique sans passer par un retour subjectif ?

Ces différents points illustrent les principales difficultés que la saisie de texte invite à considérer pour étudier l'utilisabilité d'un système. Pour mesurer la correspondance de l'étude à ces définitions, deux types de critères sont disponibles : les critères quantitatifs et les critères qualitatifs. Les critères quantitatifs sont plus aptes à proposer des réponses à des questions précises (« *la vitesse de frappe de la méthode X est plus élevée que la méthode Y* »), tandis que les critères qualitatifs donnent des réponses plus générales à des questions plus larges, mais aussi très importantes (confort, facilité d'utilisation, impression de l'utilisateur, etc.).

Dans la section suivante, nous allons donc présenter plus en détail les principales mesures qui peuvent être prises en compte pour évaluer l'utilisabilité d'un système de saisie de texte.

1.2 Mesures

1.2.1 Mesures de vitesse

La vitesse de saisie d'un utilisateur se mesure par le nombre de caractères ou de mots qu'il saisit au cours du temps.

1.2.1.a Mesure au niveau du caractère La mesure la plus simple est une mesure au niveau du caractère. Il s'agit simplement de mesurer le nombre de caractères qu'un utilisateur est capable de saisir par seconde [1]. Cette mesure est alors une vitesse en Caractères Par Seconde (CPS). Pour calculer la vitesse en CPS, lors d'une expérimentation contrôlée, on mesure simplement l'écart temporel entre la frappe du premier caractère et celle du dernier caractère. Cette mesure peut se faire au niveau du mot, ou plus habituellement au niveau de la phrase. S'agissant du temps pour mesurer un nombre de caractères effectivement tapés, on ne tient pas compte du premier caractère lors de la mesure. La formule de calcul du CPS est donc la suivante :

$$V_{CPS} = \frac{\text{nombre de caractères} - 1}{\text{temps}}$$

1.2.1.b Mesure au niveau du mot Il est possible de rendre la mesure plus applicable à la saisie quotidienne et utiliser une unité typographique un peu plus haute. Il est ainsi possible de parler de « *mots* » au sens dactylographique : en employant la définition standard [Gentner et al., 1983] du mot comme « *groupement de cinq caractères quelconques (lettres, chiffres, ponctuation ou espaces)* », nous pouvons exprimer la vitesse en « *Mots Par Minute* » (MPM). Cette vitesse peut donc s'exprimer comme :

$$V_{MPM} = \frac{V_{CPS} \times 60}{5}$$

Un utilisateur entraîné à une frappe dactylographique professionnelle tape entre 50 et 70 mots par minute en moyenne, tandis qu'il est possible d'atteindre une vitesse de 150 mots par minute pour certaines dispositions de clavier [Hempstalk, 2006].

1. http ://www.yorku.ca/mack/RN-TextEntrySpeed.html

1.2.2 Mesures de l'erreur

La mesure de l'erreur (ou de l'exactitude, les deux mesures étant complémentaires) est plus complexe. Plusieurs types d'erreurs sont possibles et parfois difficilement distinguables sans une explication de l'utilisateur sur ce qu'il a souhaité faire (ce qui est rarement le cas). L'analyse *a posteriori* est ainsi une étude extrêmement complexe. En voici les principaux éléments.

1.2.2.a Vocabulaire sur les erreurs Pour parler des différents critères qu'il est possible d'étudier lors de la saisie, nous commençons par définir certains indicateurs particuliers. Pour illustrer les différents cas d'erreurs possibles, nous allons nous baser sur l'exemple suivant où l'utilisateur a pour consigne de saisir la phrase suivante : les pyjamas du fakir.

Ce texte que l'utilisateur doit saisir, sera appelé par la suite « *texte modèle* ».

La succession de frappes réalisée par l'utilisateur pour saisir ce texte, par exemple : `[l]` `[e]` `[s]` `[ ]` `[p]` `[y]` `[j]` `[a]` `[m]` `[a]` `[s]` `[ ]` `[d]` `[u]` `[ ]` `[f]` `[a]` `[e]` `[←]` `[k]` `[i]` `[r]` sera appelée la « *saisie effective* » de l'utilisateur. Dans l'exemple ci-dessus, on peut y voir une erreur de saisie de l'utilisateur (ajout d'un caractère 'e' après le 'a' de "fakir"), immédiatement corrigée par l'utilisateur avec l'utilisation du retour en arrière ([←]).

Le texte obtenu une fois la saisie terminée, ici les pyjamas du fakir, sera appelé le « *texte obtenu* ».

Pour résumer, la saisie effective représente ce que l'utilisateur a tapé sur le clavier (la succession de touches), et le texte obtenu représente le texte réellement affiché une fois la saisie de l'utilisateur terminée. Ainsi, pour obtenir le mot fakir, l'utilisateur aurait pu saisir `[f]` `[a]` `[e]` `[←]` `[k]` `[i]` `[r]` (faekir, corrigé en fakir par effacement du e inséré par erreur) ou `[f]` `[k]` `[←]` `[←]` `[f]` `[a]` `[k]` `[i]` `[r]` (fkir, corrigé en fakir par effacement du k et ajout du a oublié par erreur), par exemple.

1.2.2.b Types d'erreur On distingue quatre grandes catégories d'erreurs possibles [Gentner et al., 1983] :

1. **omission** : une erreur d'omission apparaît dans un texte s'il manque une lettre dans un mot. Par exemple, l'utilisateur devait saisir pyjamas et a obtenu pjamas, omettant le caractère y.
2. **addition** : une erreur d'addition apparaît dans un texte s'il y a une lettre en trop dans un mot. Par exemple, fakir devient faakir, avec l'ajout d'un caractère a.
3. **substitution** : une erreur de substitution apparaît dans un texte s'il y a une lettre différente dans un mot. Par exemple, pyjamas devient pajamas, avec le remplacement du y par un a.
4. **transposition** : une erreur de transposition apparaît dans un texte s'il y a deux lettres proches échangées dans un mot. Par exemple, fakir devient fkair, avec la transposition entre les caractères a et k.

1.2.2.c Taux d'erreurs La mesure d'erreur la plus simple à calculer est la mesure d'erreur "au texte", qui correspond au nombre de caractères erronés sur le texte obtenu par rapport au texte modèle. Ce nombre est ramené en un pourcentage représentant l'erreur complète, c'est à dire la différence relative entre les deux chaînes de caractères (la chaîne de caractères produite et la chaîne de caractères souhaitée). Le taux d'erreurs est définie le plus souvent comme :

$$Tx_{erreur} = \frac{\text{nombre de caractères erronés}}{\text{nombre de caractères au total}} \times 100$$

De même, le taux de caractères correctement saisi est obtenu avec la formule suivante :

$$Tx_{correct} = \frac{\text{nombre de caractères corrects}}{\text{nombre de caractères au total}} \times 100$$

Sachant que $Tx_{correct} + Tx_{erreur} = 100\%$

1.2.2.d Taux d'erreurs sur la distance de chaîne minimale Le taux d'erreurs tel que présenté ci-dessus ne prend pas en compte les différents type d'erreurs. Il se contente de comparer le texte modèle avec le texte obtenu. Le nombre d'erreurs

peut alors être mal calculé. Une erreur de transposition peut ainsi être comptée comme deux erreurs.

Pour remédier à ce problème, il est nécessaire de mesurer la similarité entre les deux chaînes de caractères (le texte modèle et le texte obtenu). La mesure de similarité la plus connue est la distance minimale interchaîne (Minimum String Distance, ou MSD)[Ristad and Yianilos, 1998], aussi appelée distance de Levenshtein[Levenshtein, 1966]. Cette mesure permet de calculer le nombre de manipulations d'édition de caractères (suppressions, insertions, et remplacement) minimal nécessaire pour passer d'une chaîne à une autre. Par exemple, la MSD entre CHEN est CHIEN est de 1 : il faut ajouter un I entre H et E pour obtenir deux chaînes de caractères identiques. De même, entre CHIENS et CHIEN, la MSD est de 1 : il faut enlever le S après le N.

Cette mesure, en saisie de texte, permet de savoir avec quelle exactitude un texte a été recopié, en comparant la distance entre le texte modèle et le texte saisi par l'utilisateur : plus la distance minimale interchaîne est petite, meilleure est la saisie de texte.

A partir de cette mesure, il est possible de calculer un taux d'erreurs sur la distance minimale interchaîne (minimum string distance error rate, en anglais[Arif and Stuerzlinger, 2009]) :

$$Tx_{MSD} = \frac{MSD(M,O)}{max(|M|,|O|)} \times 100$$

où

— M correspond au texte modèle ;

— O correspond au texte obtenu suite à la saisie de l'utilisateur ;

— $|M|$ et $|O|$ à leur longueur respective.

L'utilisation de la plus grande chaîne de caractères assure qu'il n'est pas possible d'avoir un taux d'erreurs supérieur à 100%.

Les travaux de Soukoreff et MacKenzie[Soukoreff and MacKenzie, 2003] ont permis d'affiner la mesure de l'erreur en tenant compte de la longueur des transformations : c'est à dire, la transcription des différences entre le texte modèle et le texte

obtenu. Ainsi, pour chaque proposition de couple de texte modèle/obtenu, il y a plusieurs séries de transformations possibles, chacune expliquant de manière idoine un comportement particulier. Par exemple, en partant de fakir pour arriver à fikir, il y a plusieurs possibilités : passer de fakir à faikir (addition) puis à fikir (suppression), passer de fakir à fikir (transposition). Ces trois transformations ayant le même coût, il est difficile de pouvoir en choisir une au risque de mal représenter la saisie de l'utilisateur.

Le taux d'erreurs corrigé a permis de trouver une nouvelle formule tenant compte de ces variations :

$$TxNormalisee_{MSD} = \frac{MSD(M,O)}{\overline{S_A}} \times 100\%$$

où

- M correspond au texte modèle ;
- O correspond au texte obtenu suite à la saisie de l'utilisateur ;
- $\overline{S_A}$ correspond à la longueur moyenne des alignements possibles des différentes transformations pour l'ensemble.

Cette mesure est préférable à l'ancienne, et donne des valeurs plus intéressantes. En effet, comme la valeur moyenne est nécessairement plus grande que les transpositions unitaires, on a nécessairement $\overline{S_A} \geq max(|P|, |T|)$.

1.2.2.e Mesure de correction de l'erreur Les mesures que nous venons de présenter ne traitent que d'une partie des erreurs. En effet, seules les erreurs encore présentes dans le texte sont prises en compte. En revanche, les erreurs corrigées par l'utilisateur en cours de saisie ne sont pas prises en compte dans les mesures précédentes. Or, ces erreurs et leur gestion peuvent avoir un impact important sur la vitesse de saisie de l'utilisateur car cela requiert plusieurs actions de sa part qui au final ne produisent pas ou peu de texte supplémentaire.

En reprenant l'exemple les pyjamas du fakir, on peut le décomposer comme suit :

- Texte modèle : les pyjamas du fakir
- Saisie de l'utilisateur : `[l] [e] [s] [ ] [p] [y] [j] [e] [←] [a] [m]`

`[a] [s] [ ] [d] [u] [ ] [f] [a] [k] [i] [r]`

— Texte obtenu : les pyjamas du fakir

Ici, l'utilisateur a inséré un caractère incorrect (le "`e`") qui a été corrigé par l'appui de la touche retour arrière (le "←"). Ces frappes n'apparaissent pas dans le texte obtenu, rendant ce dernier sans faute. Le taux d'erreurs normalisé sur la distance de chaîne minimale renvoie également 0%. Cependant, la correction de cette erreur a nécessité du temps à l'utilisateur. Avec les mesures présentées jusqu'à présent, ceci n'est à aucun moment pris en compte.

1.2.2.f Nombre d'actions effectives par caractère On définit le Key Stroke Per Character (KSPC) comme étant le nombre de frappes (ou le nombre d'actions) nécessaires à effectuer pour entrer un caractère ; soit sur un texte saisi, le nombre de frappes totales par rapport au nombre de caractères entrés. On le calcule en comptant le nombre de frappes réalisées par l'utilisateur, et en le divisant par la longueur du texte obtenu :

$$KSPC = \frac{|Saisie|}{|TexteFinal|}$$

Dans le cas d'un clavier standard, le KSPC est de 1 s'il n'y a pas eu de fautes corrigées. Par exemple, sur un clavier classique physique, écrire `a` nécessite une seule action : l'appui de la touche `a`. En revanche, si l'utilisateur a effectué des erreurs qu'il a corrigées, le KSPC sera supérieur à 1. Par exemple, si l'utilisateur a frappé `z`, puis corrigé son erreur ← puis tapé la bonne lettre `a`, le KSPC est de 3, alors que le texte effectif n'est que le caractère `a`.

Pour l'exemple sur un clavier classique, avec le texte précédemment choisi, l'utilisateur a frappé 22 touches (deux pour la correction, en appuyant sur "`a`" et sur "←"), tandis que le texte obtenu n'en contient que 20. Cela nous donne un KSPC de $\frac{22}{20} = 1.1$. Si le texte avait été entré sans erreur, et sans correction, nous aurions obtenu un KSPC de 1. Par conséquent, plus il y a d'erreurs et de corrections réalisées, plus le KSPC est élevé.

Au-delà des erreurs, le KSPC peut aussi mettre en évidence la complexité à saisir

un caractère. En effet, sur certain système de saisie, la saisie d'un caractère nécessite une succession d'actions. Par exemple, avec les claviers téléphoniques à 12 touches, 3 ou 4 caractères sont liés à une même touche. Pour en sélectionner un, certaines techniques utilisent plusieurs appuis sur la touche pour lever l'ambiguïté sur la saisie du caractère. Dans ce cas, un caractère peut nécessiter 2 ou 3 frappes sur une touche pour être saisi. Le KSPC pour ces systèmes est alors supérieur à 1.

Il y a trois problèmes liés à l'utilisation du KSPC et à son application dans le domaine de l'évaluation des erreurs de saisie.

Tout d'abord, le KSPC est calculé en combinant deux valeurs ; et rend leur séparation impossible : c'est le coût factuel des erreurs commises, mais aussi, dans le cas des erreurs corrigées, de la correction qu'elles engendrent. Un KSPC élevé peut ainsi indiquer que beaucoup d'erreurs ont été effectuées mais rapidement corrigées car elles nécessitaient peu de frappes ; ou que peu d'erreurs ont été commises, mais qu'elles nécessitaient beaucoup de frappes pour les corriger. Par exemple, dans le cas où l'erreur commise est repérée tardivement par l'utilisateur, celui-ci va effectuer beaucoup d'actions pour aller la corriger. Il n'y a aucun moyen de séparer ces deux possibilités uniquement par l'observation du KSPC.

Ensuite, le KSPC dépend de la méthode employée pour saisir du texte : ainsi, un clavier tactile aura un KSPC différent [Hoggan et al., 2008] d'un clavier multifrappes [MacKenzie, 2002b]. De ce fait, pour étudier les erreurs de façon efficace entre deux méthodes de saisie, il n'est pas possible de comparer uniquement le KSPC.

Enfin, bien qu'il y ait une relation inverse entre le KSPC et le taux d'erreurs sur la distance de chaîne minimale, il est impossible de combiner ces deux mesures pour produire une valeur unifiée. Pour reprendre l'exemple de "`pyjamas du fakir`" cité plus haut, il y a un taux d'erreurs normalisé sur la distance de chaîne minimale de 0%, tandis que le KSPC est de 1.1. Si l'utilisateur n'avait pas remarqué l'erreur et évité toute correction, le taux d'erreurs normalisé sur la distance de chaîne minimale aurait été de 5%, tandis que le KSPC serait de 1.0. De plus, dans ce second cas, la vitesse aurait certainement été meilleure car l'utilisateur aurait effectué moins d'actions de correction.

Il est donc plus utile de présenter le KSPC avec le taux d'erreurs sur la distance

de chaîne minimale au niveau des résultats.

1.2.2.g Typologie des frappes Selon Soukoreff et MacKenzie, il est nécessaire de procéder à une déconstruction du processus de frappe en plusieurs catégories, plutôt que d'avoir une dichotomie brutale "saisie/demande". Ils proposent quatre catégories :

1. les frappes **correctes** (notées C) ;
2. les frappes **incorrectes, et non corrigées** (notées INF) ;
3. les frappes **incorrectes, mais corrigées** (notées IF). Les frappes IF sont celles qui ne sont pas dans le texte final mais qui sont dans la saisie effective, sans être des touches de correction ;
4. les frappes **de correction** (notées F). Les frappes de la catégorie F sont des frappes facilement identifiables : elles sont présentes comme fonctions de correction (retour arrière, suppression, déplacement du curseur) lorsqu'elles sont utilisées.

Selon cette taxonomie, le texte obtenu contient uniquement des frappes des catégories C et INF. Il est possible d'identifier les frappes INF en analysant la fonction de MSD. Les frappes C sont présentes dans le texte obtenu.

Avec une telle description des frappes, Soukoreff et MacKenzie redéfinissent les formules de calcul des erreurs comme :

$$Tx_{erreur} = \frac{INF}{C + INF} \times 100$$

$$KSPC = \frac{C + INF + IF + F}{C + INF}$$

1.2.2.h taux d'erreurs total Ultimement, la typologie de Soukoreff et MacKenzie permet de définir un taux d'erreurs total de façon assez intuitive :

$$TauxErreur = \frac{INF + IF}{C + INF + IF} \times 100\%$$

Si aucune erreur n'est commise, alors INF et IF sont tous les deux nuls, ce qui donne un taux d'erreurs total nul également.

Il est également possible de séparer ces taux d'erreurs en deux de la façon suivante :

$$TauxErreursNonCorrigees = \frac{INF}{C + INF + IF} \times 100$$

et

$$TauxErreursCorrigees = \frac{IF}{C + INF + IF} \times 100$$

avec comme propriété :

$$TauxErreur = TauxErreursNonCorrigees + TauxErreursCorrigees$$

Soukoreff et MacKenzie font la remarque que ces valeurs correspondent respectivement au MSD et au KSPC, confirmant ainsi leur pertinence.

1.2.3 Mesures de prédiction

Pour améliorer les performances de saisie de texte, beaucoup de systèmes de saisie de texte logiciels sont couplés à un système de prédiction. Pour mesurer leur efficacité, il existe 3 principales mesures [Garay-Vitoria and Abascal, 2006]. Ces mesures sont adaptées à une seule proposition de prédiction. Bien évidemment, la prédiction est plus efficace si plusieurs propositions sont faites en même temps, mais atteint rapidement un maximum asymptotique [Copestake, 1997]. De plus, en augmentant le nombre de propositions, la probabilité d'une prédiction correcte augmente. Cependant, en vertu de la loi de Hick-Hyman [Hick, 1952, Hyman, 1953], le temps nécessaire à la sélection augmente : il n'est donc pas très utile d'augmenter excessivement le nombre de propositions, cela nuit au temps de sélection, et diminue grandement l'avantage de la prédiction.

1.2.3.a Taux de succès La première mesure de l'efficacité d'un système de prédiction est le « *taux de succès* » (hit ratio) du système [Garay-Vitoria and Abascal,

2006]. Ce taux est calculé de la façon suivante :

$$TauxSucces = \frac{\#MotPredit}{\#Mots}$$

Où $\#MotPredit$ représente le nombre de mots présents dans le texte et prédits par le système de prédiction, et $\#Mot$ est le nombre total de mots représentant le texte.

En général, un bon système de prédiction obtient un taux de succès élevé, vu que le nombre de sélections nécessaires diminue. Cependant, le taux de succès n'est pas suffisant pour évaluer la qualité de la prédiction, étant donné qu'il est possible d'avoir une prédiction correcte, mais située en fin de mot, rendant la prédiction très faible quant au nombre de frappes économisées. Dans ce cas de figure, malgré un taux de succès élevé, le gain en temps pour l'utilisateur sera faible, voire même coûteux. En effet, si l'utilisateur passe trop de temps à regarder la liste des prédictions sans pouvoir l'utiliser (du fait de l'absence du mot souhaité), cela pénalise sa saisie sur le clavier et réduit sa vitesse de saisie de texte.

1.2.3.b Frappes économisées Pour permettre d'évaluer si un système de prédiction propose les bons mots rapidement lors de la saisie, il est également possible de calculer le nombre de frappes économisées lors de la saisie par la formule suivante :

$$FrappesEconomisees = 1 - \frac{\#FrappesCaracteres}{TotalFrappes}$$

Dans une sélection directe, chaque frappe produit un caractère. Il est de ce fait très facile de calculer le nombre de caractères économisés de façon analogue. Plus le mot à saisir apparaît rapidement dans la liste du système de prédiction et moins l'utilisateur a de caractères à entrer. Ainsi si le taux de frappes économisées se rapproche de 1, cela veut dire que le système de prédiction de mots est efficace et propose très rapidement le bon mot à l'utilisateur.

1.2.3.c Temps économisé Enfin, il est également possible de calculer le temps gagné de façon analogue :

$$TempsSauve = 1 - \frac{TempsMis}{TempsNecessaire}$$

Cette dernière mesure est la plus difficile à évaluer, car elle nécessite d'évaluer théoriquement le temps nécessaire pour saisir le texte sans l'aide d'un système de prédiction. De ce fait, le temps gagné par la prédiction est proportionnel aux frappes économisées.

1.2.3.d Autres mesures Pour certains, comme Greenberg, la mesure de la qualité d'un système de prédiction ne se résume pas à ces mesures. Selon Greenberg[Greenberg et al., 1995], il est en effet possible de résumer l'intégralité des mesures supplémentaires en un seul type de calcul :

Économies = Coût de la prédiction - Coût des événements standards

Ce calcul suffit en effet à résumer l'intégralité des calculs d'économie.

Prenons par exemple le temps de prédiction : l'économie de temps est évaluée en trois phases. Tout d'abord, l'élément « coût de la prédiction » correspond au temps de recherche mis par le système prédictif pour proposer la suggestion, et au temps nécessaire pour que l'utilisateur reconnaisse la proposition et la valide. Le « coût des événements standards » correspond au temps nécessaire pour l'utilisateur et le système, hors prédiction, de saisir la même chose (recherche des touches, frappe, réflexion, etc.). La mesure du gain est alors comptée comme le gain de temps de l'un en différence du second.

Il existe d'autres méthodes de classification. Par exemple, Even-Zohar et Roth[Even-Zohar and Roth, 2000], c'est le taux d'erreurs par mots et les erreurs de prédiction qui sont utilisés pour définir la performance de la prédiction : il est question de ce qui est appelé « ensemble de confusions », ce qui correspond à l'ensemble de mots possibles pour chaque saisie de l'utilisateur. L'efficacité de la prédiction est donc

modulée par deux aspects : l'adéquation de la prédiction au contexte (choisi par l'algorithme), et la taille de l'ensemble de confusions. D'après leur étude, plus l'ensemble de confusions est faible, meilleure est la déduction.

1.2.4 Mesures de la satisfaction utilisateur

Nous l'avons vu précédemment, les mesures qualitatives sont essentielles pour répondre entièrement à la question de l'utilisabilité du dispositif testé. La norme ISO 9241-11[International Organization for Standardization, 1998] définit l'utilisabilité par trois critères : efficacité, efficience et satisfaction. Comme nous l'avons montré précédemment, l'efficacité peut être évaluée par des critères quantitatifs et directement mesurables. Nous pouvons ainsi mesurer la vitesse à laquelle saisit un utilisateur, ainsi que la précision qu'il obtient avec un tel système. L'efficience peut aussi en partie être évaluée par des métriques quantitatives : par exemple en prenant en compte l'évolution de l'utilisateur avec le système. Cela montre ainsi la facilité de prendre en main le système. Cependant, pour les deux autres classes de mesures (effort et satisfaction), les mesures utilisées varient largement. En conséquence, il est très difficile de faire des comparaisons efficaces entre divers systèmes. Ce n'est pas parce qu'une fonction spécifique d'un système est efficace qu'elle le sera pour un autre. En réponse à ces contraintes plusieurs mesures qualitatives basées sur des questionnaires présentés aux participants permettent d'avoir un ensemble d'éléments pour évaluer l'efficience et la satisfaction.

1.2.4.a Outil de mesure de l'expérience utilisateur Le premier outil permettant d'évaluer l'expérience utilisateur est le questionnaire AttrakDiff[Hassenzahl et al., 2003], élaboré par Hassenzahl. Ce questionnaire permet d'évaluer un système selon des qualités et des critères hédoniques et pragmatiques. Selon le modèle de Hassenzahl, les utilisateurs perçoivent les interfaces selon deux principes : d'une part, les principes pragmatiques, c'est-à-dire ceux qui orientent l'utilisateur vers la satisfaction de la tâche, la structure, la clarté, l'efficacité ou le contrôle ; et d'autre part, les principes hédoniques, qui permettent de favoriser le bien-être de l'utilisateur au sein

de l'interface, en focalisant l'utilisateur sur des aspects de créativité, d'originalité ou de présentation. Pour Hassenzahl, ces principes modifient la perception que l'utilisateur a de l'interface, et donnent naissance à plusieurs comportements résultants d'une appréciation plus ou moins nuancée de l'interface : joie, frustration, colère, etc.

Pour ce faire, le questionnaire se présente en lui-même selon quatre sous échelles de 7 éléments, pour un total de 28. Les grandes catégories visées par AttrakDiff sont le pragmatisme (utilité de l'interface et aisance de l'utilisateur), l'hédonisme pour la stimulation (force de l'interface à maintenir une stimulation chez l'utilisateur), l'hédonisme pour l'identification (facilité de l'utilisateur à s'identifier en tant que tel), ainsi que l'attractivité globale (selon les échelles pragmatiques et hédoniques). Chaque échelle présente plusieurs différenciateurs (paires de mots à contraster) sur une base de Likert en sept points. Les éléments sont dans un ordre standardisé, mélangé, et jamais passés successivement. Le score obtenu permet de formuler une vision globale de l'expérience utilisateur.

1.2.4.b Mesure de l'utilisabilité Une échelle de satisfaction simple dans l'évaluation d'interfaces est le questionnaire SUS [Brooke et al., 1996]. Celui-ci permet de projeter le choix des utilisateurs suivant certains préceptes. Une échelle Likert est utilisée pour indiquer sa concordance avec les préceptes en question.

Les principes du questionnaire SUS sont typiques d'une échelle Likert : des exemples extrêmes de situations ou d'énoncés en rapport avec le sujet d'étude sont pris pour forcer un positionnement de l'utilisateur en fonction de ces derniers. Le questionnaire SUS a été conçu de cette façon, en amalgamant des commentaires de plusieurs questionnaires initiaux sur l'utilisabilité, répartis sur plusieurs groupes significatifs d'utilisateur, pour obtenir un questionnaire de dix énoncés tous notés avec une échelle de 1 à 5.

Ce questionnaire est rempli par l'utilisateur une fois l'évaluation terminée, mais avant toute discussion ou information avec l'expérimentateur quant au système. Les réponses données doivent être immédiates, sans réflexion préalable sur une question. En cas d'hésitation, le score médian est donné à la question en cours.

Le questionnaire SUS renvoie ensuite un score unique sur 100, représentant une

mesure combinée de l'aisance d'utilisation d'un système. Les scores des éléments individuels du SUS n'ont aucune signification en dehors. Plus le score SUS est élevé, meilleur est le système. Le score d'utilisabilité minimal classiquement utilisé est de 70 (acceptable). Nous présentons la version traduite du questionnaire utilisée en partie annexe.

1.2.4.c Mesure de la charge cognitive L'obtention d'une mesure de la difficulté d'une tâche ou de la charge de travail nécessaire durant le déroulement d'une tâche n'est pas facile. Or, il est crucial de noter que la charge de travail perçue par un utilisateur peut affecter la performance de la tâche ; que cela soit par un stress mental excessif, ou, au contraire, nul. Il est de ce fait nécessaire de mesurer effectivement la surcharge de l'utilisateur au cours de la tâche. Un des moyens les plus efficaces pour ce faire est un test similaire au SUS, de par les caractéristiques qu'il possède : le test doit être sensible, permettre un rapide diagnostic des caractéristiques propres de l'utilisateur lors de la tâche, facile à implémenter sur tout support, et bien sûr, être fiable.

Le test NASA-TLX (pour Task Load Index), a été conçu suite aux travaux du Human Performance Group associé au Ames Research Center de la NASA[Human Performance Group NASA Ames Research Center, 1986] en 1986. Ce test permet d'obtenir une estimation de la charge de travail d'un utilisateur lors de la manipulation d'un système, ou une fois son utilisation terminée. À nouveau, ce test permet de mettre en évidence et de mieux cerner les critères de confort, de satisfaction, d'efficacité et de sécurité lors de l'utilisation de systèmes. Ce test se base sur six principales valeurs pour évaluer la charge de travail : l'exigence mentale, l'exigence physique, l'exigence temporelle, la performance, l'effort et la frustration. À chacune des valeurs analysées, le NASA-TLX présente une double échelle à 20 incréments, attribuant ainsi un score entre 0 et 100 (arrondi au cinquième) pour chaque critère. Utiliser une double échelle limite la variabilité entre les sujets tout en étant plus précis. Une pondération des six critères permet de calculer le score de charge totale. L'hypothèse mesurée par le NASA-TLX [Hart and Staveland, 1988] est de considérer que la charge de travail est centrée sur l'humain et non sur la tâche. De

ce fait, la charge de travail résulte de l'interaction entre les demandes de la tâche, des circonstances de son exécution et des capacités, comportements et perceptions de l'utilisateur.

En comparaison avec des méthodes de mesure de la charge de travail, l'approche subjective proposée par le groupe Ames est la plus proche de ce qu'il est possible de mesurer : en effet, il y a deux facettes quant à la charge de travail. D'une part, les perceptions immédiates apparaissant spontanément à l'utilisateur ; et d'autre part, une évaluation provoquée par les prérequis de l'expérience. L'utilisation du NASA-TLX permet d'obtenir des résultats rapides (environ moins d'une minute pour remplir les six échelles demandées suite à une condition expérimentale).

Comparé à d'autres méthodes, le NASA-TLX est moins intrusif, facile à implémenter et portable pour des expériences à distance et est un des outils les plus utilisés en recherche [Cao et al., 2009]. L'approche du NASA-TLX pour produire un score évaluant la charge de travail à partir de six axes principaux donne une bonne indication qui justifie son utilisation dans le cadre de notre travail.

2 Systèmes de saisie de texte sur dispositifs mobiles

Comme nous l'avons indiqué dans notre introduction, les dispositifs mobiles sont de plus en plus utilisés pour une communication écrite. Avec la multiplication des smartphones et l'apparition des écrans tactiles qui permettent une approche dynamique, les supports physiques comme les claviers à touches sont en voie de disparition, remplacés par des solutions de saisie alternatives.

Nous présenterons les différents types de systèmes existants sur dispositif mobile avant de nous focaliser ensuite sur les systèmes de saisie adaptés aux déficients visuels.

2.1 Systèmes de reconnaissance de parole

Une première approche consiste à utiliser un système de reconnaissance de la parole. Selon Azenkot [Azenkot and Lee, 2013], les déficients visuels qui n'utilisent que peu une saisie gestuelle, ont une préférence pour les systèmes similaires à Siri, sur IOS, ou Google Now, sur Android. Ces systèmes proposent une saisie de texte à partir d'un système de reconnaissance vocale [Shneiderman, 2000]. Le principe est simple : lorsqu'il est invité, l'utilisateur dicte directement ce qu'il souhaite taper au téléphone, qui le retranscrit directement sur l'application qu'il utilise (réseau social, SMS, etc.). Pour l'utilisateur en situation de déficience visuelle, cela évite de manipuler un système qu'il peut difficilement contrôler comme un système de saisie sur écran tactile. Cette approche se veut également plus simple en termes de fatigue : il n'y a pas besoin de chercher une interaction particulière, de parcourir l'écran tactile pour saisir chaque caractère, ou apprendre un nouvel alphabet ou une nouvelle méthode de saisie. Cependant, la saisie par reconnaissance vocale pose deux principaux problèmes.

Le premier obstacle que cela pose vient du manque de discrétion lors de l'utilisation d'un tel système car toute personne à proximité de l'utilisateur peut alors entendre et suivre ce qu'il est en train de faire. De ce fait, la vie privée de l'utilisateur est très diminuée. Ceci peut ainsi être un frein à l'utilisation d'un tel système pour de nombreux utilisateurs.

Le second problème est le taux de reconnaissance de parole de tels systèmes. En théorie, ces systèmes ont un bon taux de reconnaissance de la parole. En revanche, pour peu que la saisie ait lieu dans un environnement bruité (soit dans un milieu bruyant, soit par un problème d'élocution de l'utilisateur, soit à cause d'un périphérique dysfonctionnel), le taux de reconnaissance diminue très rapidement, ce qui rend alors le système de reconnaissance inutilisable pour effectuer de la saisie de texte.

2.2 Systèmes de reconnaissance de forme

La seconde classe de techniques concerne les systèmes basés sur la reconnaissance d'écriture et plus généralement de forme. Ces systèmes étaient plus couramment

utilisés à l'apparition des appareils mobiles dont l'interaction avec l'écran se faisait au moyen d'un stylet.

L'exemple le plus connu, pour la saisie à base de reconnaissance de forme est l'alphabet Graffiti [MacKenzie and Zhang, 1997] (figure 2) : ce système se base sur l'utilisation de "raccourcis d'écriture". Ces raccourcis sont des monographes, traçables d'un seul trait, destinés à être écrits sans regarder l'écran. L'utilisateur les dessine sur une zone tactile à l'aide d'un stylet. Comme les monographes sont proches des caractères de l'alphabet réalisés en écriture manuscrite, cette grammaire de gestes nécessite peu d'apprentissage de la part des utilisateurs.

FIGURE 2 – En haut, l'alphabet Graffiti ; en bas, Unistroke.

Un autre alphabet, appelé Unistroke (figure 2) a été conçu sur le même principe que Graffiti, mais en simplifiant les gestes utilisés : les traits de l'alphabet sont facilement distinguables et donc plus simples à réaliser. La simplification de chaque geste a pour but d'augmenter le taux de reconnaissance par le système. Cela permet ainsi d'augmenter la vitesse de saisie de texte et surtout de réduire le taux d'erreur de reconnaissance par rapport à l'alphabet Graffiti [Castellucci and MacKenzie, 2008]. Cependant, malgré cette simplification des gestes, le taux d'erreur dû à une mauvaise reconnaissance des gestes de l'utilisateur est encore trop élevé et ce même après plusieurs sessions d'utilisation [Castellucci and MacKenzie, 2008].

Pour réduire ce problème de reconnaissance, des alphabets de geste sont basés sur des points de repère précis. Par exemple, EdgeWrite[Wobbrock et al., 2003] utilise les 4 coins d'un carré comme points de repère. Chaque geste représentant un caractère est alors une séquence constituée à partir de ces 4 coins sur lesquels l'utili-

sateur doit déplacer le stylet ou le doigt. Une surface rectangulaire en plastique peut être ajoutée sur le smartphone et ainsi faciliter la saisie pour les utilisateurs dont le contrôle moteur n'est pas précis ou difficilement maîtrisé :l'utilisation de bords rigides[Wobbrock et al., 2003] comme guide permet d'effectuer ces gestes plus rapidement et plus précisément, contrairement aux systèmes de reconnaissance de geste comme Graffiti.

Cependant, malgré l'avantage de permettre une saisie proche de l'écriture "naturelle", les systèmes à base de reconnaissance de formes pèchent par deux principaux désavantages : d'une part, la reconnaissance de la forme est primordiale. Si la forme est mal saisie, l'utilisateur devra la ressaisir. D'autre part, l'alphabet utilisé nécessite une phase d'apprentissage des différents gestes constituant l'alphabet. Ce temps d'apprentissage amène souvent les utilisateurs à utiliser d'autres systèmes de saisie de texte.

Ces deux principaux inconvénients font que ces systèmes à base de reconnaissance de geste sont très peu utilisés par les déficients visuels.

2.3 Claviers logiciels

Le système de saisie de texte le plus connu et le plus utilisé est le clavier logiciel. Cependant, une des limitations principales est de proposer un clavier plus réduit qu'un clavier physique classique : en effet, l'intégralité des touches ne peut pas y être entièrement représentée sous peine de rendre la saisie impossible faute de touches trop petites ou inaccessibles. De ce fait, on est obligé de dissocier l'action de « frappe de touches » en plusieurs actions, suivant le type de caractère souhaité : lettre, ponctuation, symbole, chiffre, lettre accentuée, etc. On introduit le plus souvent une touche de bascule qui permet de changer le mode du clavier et accéder à une série de touches différentes.

Enfin, un des grands inconvénients liés au clavier logiciel est le manque de retour proposé à l'utilisateur. En effet, à la différence du clavier physique où l'utilisateur à un retour haptique sur ces actions via l'enfoncement et le relâchement des touches du clavier, il perd sur un clavier logiciel sur écran tactile cette perception des actions qu'il

réalise. Ce problème peut être atténué par des effets visuels proposés à l'utilisateur, mais cela nécessite de la part de ce dernier de regarder constamment son clavier. Ce problème devient majeur dans le cas d'une utilisation par des utilisateurs déficients visuels, où le retour visuel initialement privilégié n'est pas utilisable. Des modalités de substitution sont alors nécessaires.

Nous allons présenter dans cette section les différents types de claviers logiciels qui existent, les différentes interactions qu'ils proposent, puis nous focaliserons sur les solutions proposées pour une interaction sans regard et nous finirons sur les solutions proposées spécifiquement pour les non-voyants.

2.3.1 Optimisation de la disposition des caractères

Le plus souvent, ces claviers logiciels sont arrangés avec une disposition fixe au cours du temps ; l'interaction avec ces claviers est identique à celle utilisée sur les claviers de bureau classique : l'utilisateur frappe la zone représentant la touche qu'il souhaite actionner, et le caractère choisi est validé.

Comme la majorité des claviers logiciels reprennent l'agencement classique des claviers physiques, ils ne sont donc pas optimisés pour une saisie rapide. La disposition principale, le QWERTY (ou l'AZERTY selon la culture) est une disposition initialement conçue pour faciliter la saisie à deux mains sur les machines à écrire [Liebowitz and Margolis, 1990] : en effet, les caractères qui se succèdent fréquemment ont été placés aux deux extrémités pour d'une part faciliter l'utilisation des deux mains de manière séquentielle, mais aussi pour éviter que lors d'une saisie rapide, les bâtonnets de la machine à écrire ne s'entremêlent. Cette contrainte mécanique, a disparu des claviers physiques actuels, et bien évidemment des claviers logiciels où la saisie se fait généralement à un seul doigt (ou pointeur).

C'est pourquoi il est possible d'optimiser la saisie en proposant une disposition différente. On peut citer les optimisations au moyen d'heuristiques de résolution de problèmes complexes comme, par exemple, la disposition métropolis obtenue grâce à un algorithme de recuit simulé [Zhai et al., 2000], ou le système GAG qui permet de concevoir une disposition de caractères à partir d'algorithmes génétiques[Raynal

and Vigouroux, 2005a]. Ces algorithmes offrent la possibilité de s'adapter à la langue utilisée et ainsi proposer des dispositions optimisées en fonction de la fréquence d'utilisation de chaque caractère [Raynal, 2006].

D'autres claviers utilisent des dispositions alternatives qui sont plus adaptées à une interaction particulière, où l'intérêt n'est pas la rapidité mais la précision : ainsi, le système ROTEX utilise l'agencement ENBUD, conçu spécifiquement pour éviter les ambiguïtés lors de la frappe. Pour cela, Walmsley[Walmsley et al., 2014] introduit le concept de « caractères interchangeables » : deux caractères sont dits interchangeables entre eux s'il est possible de les remplacer l'un par l'autre dans un mot, tout en obtenant un mot existant dans le dictionnaire. Les caractères interchangeables sont alors espacés d'au moins une lettre, obtenant ainsi un agencement qui minimise la possibilité d'erreurs de frappe. L'agencement final obtenu est divisé ensuite en cinq blocs ("ENBUD", "JCOFLY", "QTHVIG", "MXRZP" et "KWAS").

2.3.2 Problématiques des petites surfaces

Dans le cas des téléphones portables ou des dispositifs portables, la majeure partie des problèmes, que cela soit de précision des actions, de taille des caractères ou de quantité d'informations affichées, vient de la petite taille de l'écran : lorsque la zone d'affichage est petite, les actions sont généralement imprécises (doigts trop gros ou zone interactive trop petite), et les informations sont moins visibles (petits caractères, formes floues). Pour pallier ce problème, différentes techniques existent.

2.3.2.a Regroupement des caractères Pour agrandir les touches du clavier, une première solution consiste à regrouper plusieurs caractères sur une même touche. Ce type de clavier est appelé clavier ambigu. Pour lever l'ambiguïté sur le caractère souhaité sur la touche, plusieurs interactions ont été envisagées. La plus connue consiste à taper plusieurs fois sur la touche en fonction de la position du caractère sur la touche : Le premier caractère sera saisi avec une seule frappe, le second avec deux frappes, etc. Cette technique dite multi-frappes initialement utilisée sur les claviers physiques des téléphones mobiles a été reprise sous forme logicielle pour

limiter le nombre de touches affichées à l'écran [Sánchez and Aguayo, 2007].

Avec un écran tactile, d'autres interactions ont aussi pu être testées comme celle consistant à utiliser un menu circulaire pour sélectionner la touche contenant le caractère souhaité [Bonner et al., 2010] (voir figure 3). L'écran est ensuite agencé pour présenter sous forme d'une liste seulement les caractères présents sur la touche sélectionnée. L'utilisateur n'a plus qu'à sélectionner le caractère souhaité. Ainsi, chaque caractère est accessible au moyen de deux actions (sélection de la touche, puis du caractère sur celle-ci).

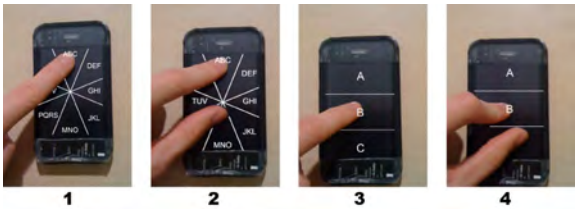

FIGURE 3 – Le système No-Look-Notes : sélection du groupe de lettres (1,2) et choix de la lettre B (3,4).

Enfin, d'autres techniques s'inspirent du clavier T9[Bi et al., 2013] pour l'appliquer aux écrans tactiles[Shahzadi et al., 2011]. Dans ce cas, les systèmes utilisent un algorithme de prédiction pour limiter le nombre d'actions nécessaires à l'utilisateur et garantir une vitesse de frappe correcte.

2.3.2.b Augmenter la taille des touches Il est également possible de changer l'apparence du clavier. Une des solutions les plus simples est de proposer un niveau de grossissement qui varie en fonction de la saisie : par exemple, le système ZoomBoard[Oney et al., 2013], fonctionnant sur montre tactile, présente un clavier intégral au format 16×6 mm. Par défaut, l'intégralité du clavier est affichée à l'écran ; et ce n'est que lorsque l'utilisateur appuie sur une touche que le clavier itérera suivant différentes tailles avant de proposer une taille adéquate. Une fois la saisie terminée, le clavier reprend sa taille initiale. De façon analogue, le système SplitBoard[Hong et al., 2015] (et son homologue Swipeboard[Chen et al., 2014]) propose un fort grossissement, divisant le clavier en plusieurs "zones" entre lesquelles l'utilisateur peut

basculer par un geste de glissement. Il n'est pas nécessaire que le grossissement soit uniforme : le clavier Fisheye[Raynal and Truillet, 2007], propose un grossissement "en lentille", centré sur une touche en particulier. Un autre clavier, DriftBoard[Shibata et al., 2016], pour les systèmes ultra-petits, utilise un déplacement relatif suivant des "drifts", le mouvement du doigt étant ralenti puis appliqué à un pointeur, de façon à éliminer tout manque de précision.

2.3.2.c Arrangement dynamique Enfin, il existe également des claviers logiciels offrant une information différente en fonction de la saisie, comme le système KeyGlass[Raynal and Vigouroux, 2005b], qui propose autour de la dernière touche frappée, les caractères les plus probables suivant un caractère venant d'être saisi. Ainsi, la distance à parcourir entre deux caractères à saisir est fortement réduite [Raynal, 2007].

De même, certains claviers peuvent s'adapter aux besoins de l'utilisateur, et modifier leur agencement, jusqu'à enlever des lettres au fur et à mesure de la saisie en diminuant l'espacement entre les touches pour augmenter la taille de ces dernières, comme SpreadKey[Merlin and Raynal, 2009], BigKey[Al Faraj et al., 2009] ou FloodKey[Aulagner et al., 2010].

Enfin, une dernière utilisation des résultats du système de prédiction de caractères consiste à changer le fonction de gain du curseur du dispositif de pointage en fonction de la probabilité d'apparition de chaque caractère [Raynal et al., 2014]. Ainsi, l'apparence du clavier ne change pas pour l'utilisateur, mais le pointeur a une vitesse de déplacement sur le clavier différente en fonction du caractère sur lequel il se trouve.

2.3.3 Interactions spécifiques

Au-delà de l'interaction standard consistant à pointer sur l'écran tactile les différentes touches du clavier logiciel, d'autres techniques d'interaction ont été testées pour saisir du texte avec un smartphone.

2.3.3.a Saisie à base de gestes sur l'écran tactile A la place d'activer les touches du clavier logiciel par des frappes sur l'écran, certains systèmes permettent de saisir du texte sur clavier logiciel au moyen de gestes effectués au dessus du clavier. La saisie s'effectue en survolant, lors du tracé, les caractères désirés.

La première méthode à avoir utilisé ce principe est QuickWriting[Perlin, 1998], qui divise l'écran en une série de 8 zones liées chacune à chaque point cardinal (Nord, Sud, Est, Ouest, Nord-Ouest, Nord-Est, Sud-Ouest et Sud-Est respectivement). Chaque lettre est alors disposée dans une des zones, selon sa probabilité. Pour saisir le mot "the" (voir figure 4), par exemple, il suffit d'effectuer le tracé vers les zones Est, Ouest, et Nord. La saisie étant continue, relâcher le doigt ou le stylet termine la saisie du mot. Ce système se veut avoir une vitesse de 18 MPM.

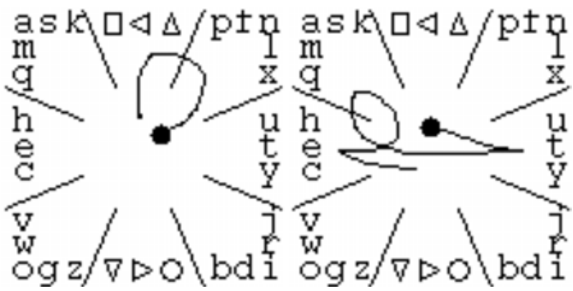

FIGURE 4 – Le clavier QuickWriting : à gauche, saisie de f ; à droite, saisie de "the".

Lorsque le geste est localisé sur l'écran, on parle plus facilement de reconnaissance de forme[Zhai and Kristensson, 2010]. Parmi les systèmes de saisie se basant sur un clavier logiciel avec une reconnaissance de formes, nous pouvons citer le système de ShapeWriter[Zhai et al., 2009]. L'utilisateur doit effectuer avec son doigt posé sur l'écran le chemin reliant tous les caractères composant le mot qu'il souhaite saisir. Une fois le doigt levé, le système réalise une déduction du mot saisi suivant la forme du tracé obtenu et de l'ensemble des touches survolées au cours de la réalisation de ce chemin.

Certains systèmes de saisie prennent avantage du caractère multitouch du clavier en permettant une saisie simultanée des deux mains, comme le Bimanual Gesture Keyboard[Bi et al., 2012] : ici, le clavier est séparé en deux zones, délimitées pour

la disposition QWERTY par les caractères de Q à T d'une part et les lettres Y à P d'autre part, ainsi que des touches situées en dessous respectivement (voir figure 5). Une telle technique permet d'être plus efficace : au lieu de déplacer le doigt sur la totalité du clavier, deux gestes, un pour chaque main, sont effectués lors de l'entrée d'un mot. Les caractères sont saisis dans l'ordre de leur apparition dans le mot. Deux validations sont possibles : soit au relâchement du doigt, soit à l'appui de la touche espace. Cependant, cette technique est plutôt destinée aux périphériques munis d'un grand écran, comme les tablettes. De plus, l'utilisation de tracés sur le clavier n'est pas forcément facile pour l'apprentissage de cette nouvelle forme de saisie (selon l'étude, seuls 42% des utilisateurs seraient prêts à l'utiliser de façon quotidienne).

FIGURE 5 – Bimanual Gesture Keyboard : saisie de « *life* » sur le clavier.

2.3.3.b Gestes réalisés avec le dispositif Les techniques à base de geste du smartphone sont des techniques de saisie qui ne se basent généralement pas sur une interaction avec l'écran, mais sur une série de mouvements que l'utilisateur réalise en bougeant le téléphone.

Le premier système connu utilisant des mouvements du dispositif est le système ROTEXT (voir figure 6), produit par Walmsley [Walmsley et al., 2014]. C'est un système prévu pour être utilisé de façon rapide, et imprécise. Utilisant l'agencement ENBUD sur une seule ligne, la saisie s'opère alors en inclinant le smartphone de gauche à droite suivant un demi cercle pour choisir les lettres, la confirmation s'effectuant par un appui sur l'écran. La validation du mot se fait par une légère secousse

sur l'écran. Un mot est alors proposé à l'utilisateur parmi les mots possibles : l'ambiguïté est traitée par ROTEXT en effectuant ensuite en comparant le mot tapé avec tous les autres mots de même longueur, mais différents d'une seule lettre. Toutes les confirmations et propositions sont offertes à l'utilisateur par un retour audio (synthèse vocale). En cas de correction nécessaire, l'utilisateur secoue le téléphone de nouveau pour passer au mot similaire suivant. Bien que manquant en moyenne 3 positions de lettres par saisie, les utilisateurs peuvent atteindre une vitesse de frappe d'environ 12 MPM avec un taux de succès de 99% après entraînement (environ 4,3 heures) ; pour un maximum de 40 mpm théorique par un utilisateur expert.

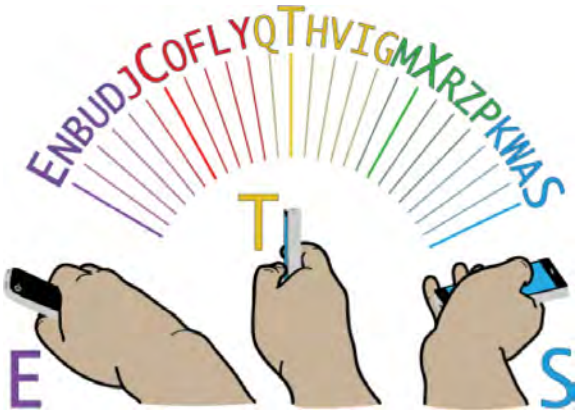

FIGURE 6 – Le système ROTEXT : l'inclinaison du téléphone associée à une secousse permet de choisir un caractère (ici, E, T ou S).

La majorité des techniques utilisant les gestes du smartphone se basent sur l'utilisation du gyroscope et de l'accéléromètre. De façon similaire à ROTEXT, il y a également le système TiltType[Partridge et al., 2002] (voir figure 7). Ce système permet également une saisie non-visuelle, et est particulièrement adapté aux petits dispositifs ou aux dispositifs ne présentant pas d'écran. Pour rentrer un caractère, l'utilisateur incline le dispositif et appuie sur des boutons. Le caractère entré dépend de l'inclinaison du téléphone et du bouton qui a été appuyé : ainsi, l'appui sur un ou plusieurs des quatre boutons, combiné avec les neuf directions possibles d'inclinaison, permet d'entrer 216 caractères.

Malgré l'avantage de ne pas nécessiter de regarder l'écran, l'utilisation de gestes

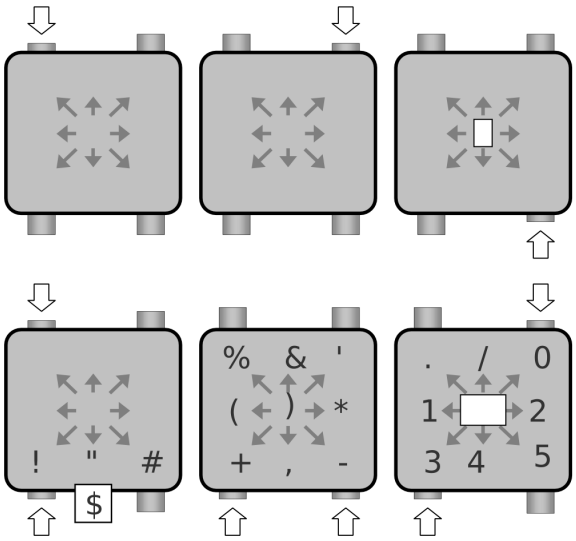

FIGURE 7 – Le système TiltType : l'appui sur les quatre boutons extérieurs combiné à l'inclinaison du dispositif permet la saisie du caractère souhaité.

pour bouger le smartphone nécessite un contrôle kinesthésique précis des muscles utilisés (avant-bras, poignet, etc.). Ces gestes sont fastidieux pour l'utilisateur, rendant une utilisation à long terme potentiellement pénible. De même, l'apprentissage d'un agencement différent mêlant une combinaison d'inclinaisons du dispositif et de pressions de boutons nécessite un certain temps d'adaptation à la technique d'interaction et d'apprentissage de la grammaire associée. Ceci n'est pas sans conséquence sur le temps et la motivation nécessaire pour utiliser ce système de saisie de texte.

2.3.3.c Gestes réalisés autour du dispositif Il existe également des systèmes comme Kinemic Wave[Amma et al., 2016], laissant une entière liberté à l'utilisateur de contrôler le dispositif et effectuer de la saisie de texte par des gestes dans n'importe quelle situation de mobilité, et sans avoir à toucher le smartphone. Cependant, ces dispositifs utilisent fréquemment des capteurs supplémentaires (comme dans une montre connectée dans le cas de Kinemic Wave), pour pouvoir détecter la position des mains par rapport au dispositif. De tels systèmes permettent d'entrer des commandes, comme le défilement ou la confirmation ; tandis que des gestes

plus complexes permettent certaines interactions plus longues, comme la recherche de termes ou l'entrée de messages et d'annotations. La saisie de texte se fait alors en écrivant le texte voulu dans les airs, sans que l'utilisateur n'ait besoin de marquer une pause. Par exemple, pour écrire, l'utilisateur place sa main gauche comme s'il tenait un bloc-notes, et trace les lettres comme s'il tenait un stylo (air writing) pour un vocabulaire limité de 10 000 mots. Un tel système permet une grande liberté pour l'utilisateur, mais n'est pas adapté au grand public, du fait de la nécessité de capteurs annexes au dispositif principal.

2.4 Claviers adaptés pour une saisie sans regard

Les écrans tactiles posent un souci de taille pour les utilisateurs en situation de déficience visuelle. Au lieu d'avoir un périphérique en relief, les utilisateurs se retrouvent avec une surface lisse. L'utilisation d'interfaces graphiques classiques sur laquelle on interagit au moyen de doigt touchant l'écran rend le problème encore plus difficile à résoudre. Il existe plusieurs approches pour permettre une saisie non visuelle sur un dispositif mobile à l'aide de claviers logiciels.

La première méthode pour permettre une saisie non visuelle avec un clavier logiciel est la plus simple et la plus connue, car elle nécessite peu d'adaptation sur la saisie de texte utilisée. En effet, cette technique est souvent appliquée en couplant une synthèse vocale avec un clavier logiciel. Ainsi, toute action effectuée sur le clavier logiciel est vocalisée : lors du passage du doigt sur l'écran, les lettres situées au-dessous sont prononcées. La validation s'effectue alors en relâchant le doigt de l'écran, et le caractère choisi est alors énoncé dans une tonalité plus aiguë.

Cette approche reprend le fonctionnement du clavier physique classique, si l'utilisateur a l'habitude d'utiliser un clavier de bureau, la saisie de texte n'est pas fondamentalement différente. En revanche, l'exploration de l'écran n'est pas du tout adaptée à une utilisation intensive du fait de problèmes de précision et de fatigue.

Un des systèmes parmi les plus couramment employés sur le marché des téléphones est VoiceOver [Leporini et al., 2012], conçu par Apple. Ce système repose sur l'utilisation d'une "boîte focus", qui est déplacée au moyen d'un doigt posé sur

l'écran vers l'élément avec lequel on souhaite interagir. Un double tap sur l'écran active l'élément courant.

En conséquence, la saisie au sein de VoiceOver se base sur ce principe. Les éléments sélectionnables sont les touches du clavier. Chaque caractère situé sous le doigt posé à l'écran est lu. Activer une touche permet de saisir le caractère lié à cette touche. Il est également possible de lire le texte entré en faisant glisser le doigt vers le haut ou vers le bas. Cependant, VoiceOver [2] oblige l'utilisateur à parcourir un grand nombre de petites cibles, rendant la localisation d'une lettre difficile pour un utilisateur qui n'a aucune notion du clavier physique ou de l'agencement des touches.

Le fait que VoiceOver soit inclus par défaut sur la totalité des iPhones permet à un grand nombre d'utilisateurs déficients visuels d'y accéder. La plupart des conseils et guides ergonomiques pour les systèmes de suppléance basés sur un clavier logiciel augmenté d'une synthèse vocale y font référence.

Au-delà d'une utilisation sur clavier logiciel standard, l'usage de la synthèse vocale a été adaptée à différents systèmes de saisie de texte pour permettre une saisie sans regard avec ces systèmes. Par exemple, certains claviers ambigus vus précédemment proposent l'usage d'une synthèse de parole pour guider l'utilisateur dans sa saisie. C'est notamment le cas des solutions « Mobile Messenger for the Blind » [Sánchez and Aguayo, 2007] et No-Look-Notes[Bonner et al., 2010].

De même, Tinwala et MacKenzie[Tinwala et al., 2009] ont développé un système en réutilisant l'alphabet de Graffiti. Ce système permet l'utilisation des doigts en sus d'un stylet. L'application proposée permet aux utilisateurs de saisir un caractère à la fois, avant de terminer la saisie par un caractère espace. Suivant le cas, le système prononce le caractère correspondant au trait qui a été reconnu (dans le cas d'une reconnaissance correcte) ou vibre une fois (pour indiquer une saisie incorrecte). Une fois le mot terminé, l'utilisateur indique la fin de sa saisie par un double-tap, correspondant au caractère espace, confirmé par un "bip" sonore du système. Un tel système permet une vitesse de 21 MPM au maximum avec un très bon retour (95% de réussite).

2. VoiceOver http ://www.apple.com/accessibility/iphone/vision.html

Il est également possible d'utiliser un système de saisie de texte pour lequel le retour audio est suffisant pour permettre à l'utilisateur d'entrer du texte et de se repérer dans sa saisie sans regarder l'écran, à l'image du système ROTEX[Walmsley et al., 2014] dont toutes les interactions sont basées sur la manipulation du smartphone en lui-même, ce qui rend donc plus complexe l'utilisation de l'écran pour afficher de l'information. En effet, lors des manipulations du smartphone par l'utilisateur, l'écran n'est plus forcément face à l'utilisateur, ce qui rend alors impossible l'affichage d'information sur celui-ci. L'utilisation de ces systèmes est cependant considérée comme une utilisation en mode "expert" : elle est habituellement destinée aux utilisateurs suffisamment habitués à les utiliser pour se passer de retour visuel, ce qui restreint leur utilisation "grand public".

Enfin, il existe des solutions dont l'interaction est radicalement différente du clavier classique, comme c'est le cas avec la solution proposée par [Azenkot and Lee, 2013]. Pour rentrer un chiffre, l'utilisateur appuie ou glisse un, deux ou trois doigts à l'écran. Les dix chiffres sont codés par une combinaison de ces gestes, en s'efforçant de représenter la sémantique du chiffre : par exemple, le chiffre 2 est saisi avec deux doigts. Cependant, cette méthode est restreinte aux seuls chiffres et n'est pas applicable pour de grands alphabets du fait du manque de points pour pouvoir coder les différents caractères.

2.5 Claviers Braille

Les claviers les plus adaptés aux utilisateurs en situation de déficience visuelle sur dispositif mobile sont pour la plupart basés sur l'alphabet Braille où les lettres sont codées par une matrice de points composée de trois lignes et deux colonnes. Chaque caractère correspondant à un arrangement particulier de ces six points.

Le clavier BrailleOne, produit par Hatzigiannakoglou[Hatzigiannakoglou and Kampouraki, 2016], s'appuie sur une division de l'écran en trois secteurs verticaux, utilisés comme un clavier Braille à une main. Chacun des trois secteurs représente les trois premiers points d'une matrice Braille (1,2,3) lors de la première frappe, puis les trois derniers (4,5,6) lors de la deuxième frappe de l'utilisateur. Pour BrailleType (figure

8) et BrailleTouch (figure 9) [Southern et al., 2012], l'écran est séparé en six cellules de même taille (trois lignes et deux colonnes). Chaque cellule correspond à un point de la matrice Braille. Pour activer les points nécessaires à la composition d'un caractère, l'utilisateur doit appuyer sur les cellules souhaitées. La validation du caractère se fait de façon automatique après un certain délai. Le système BrailleKey est lui basé sur l'utilisation de seulement deux cellules représentant respectivement la colonne de gauche et celle de droite [Subash et al., 2012]. L'activation des points des trois lignes se fait alors par des actions différentes sur la cellule correspondante à la colonne. Par exemple, le point de la première ligne de la colonne de gauche est activé par une frappe simple sur la cellule de gauche, celle de la deuxième ligne par une double frappe et celle de la troisième ligne par un appui long sur cette même cellule. Les actions sont identiques sur la cellule de droite pour activer les points de la colonne de droite.

FIGURE 8 – Le clavier BrailleType.

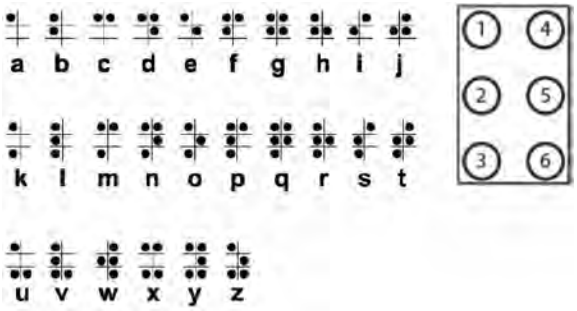

FIGURE 9 – Le clavier BrailleTouch.

D'autres solutions proposent des interactions différentes pour saisir un caractère

Braille. TypeInBraille est, par exemple, basé sur une interaction multifrappe : pour saisir un caractère, l'utilisateur doit effectuer trois multifrappes successives sur l'écran (figure 10). Chaque multifrappe correspond à une combinaison permettant d'entrer une ligne de la matrice Braille nécessaire pour la saisie du caractère. Il existe quatre combinaisons possibles : ne pas activer de point, activer le point de gauche, activer le point de droite, et activer le point de gauche et le point de droite. Sur le même principe de multifrappe, Šepić et al.[Šepić et al., 2014] proposent BrailleEasy, une solution qui imite le clavier Braille standard, mais divise la frappe d'un caractère en deux combinaisons multipoints. Par exemple, pour saisir n, l'utilisateur appuie ainsi avec l'index et l'annulaire sur la première colonne, puis avec l'index et le majeur pour entrer la deuxième colonne du caractère Braille. Si une colonne est vide (aucun des trois points n'est activé), l'utilisateur glisse à travers l'écran avec un seul doigt.

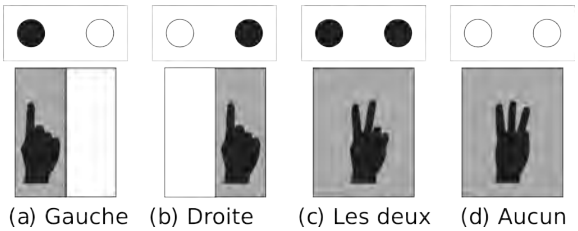

FIGURE 10 – L'approche TypeInBraille.

Il existe également des techniques à base de gestes. Par exemple, le Clavier Eyedroid[Shabnam and Govindarajan, 2014] utilise cinq modèles de gestes définis sur l'écran tactile pour permettre aux déficients visuels de saisir du texte. Les gestes enregistrés pour entrer des points Braille sur l'écran tactile sont simples : glissement de gauche à droite pour activer le point gauche et désactiver le point droit, glissement de droite à gauche pour activer le point droit et désactiver le point gauche, appui sur l'écran avec un doigt pour désactiver les deux points, et glissement de bas en haut pour activer un point. Ce clavier présente plusieurs avantages : tout d'abord, il résout le problème de navigation, car il ne nécessite pas de la part de l'utilisateur de trouver un objet dans un emplacement spécifique. Il utilise également un retour vocal pour aider les utilisateurs à interagir avec le clavier. Cependant, il n'est pas facile à

comprendre, et oblige les utilisateurs à mémoriser les fonctions gestuelles définies.

De façon similaire, EdgeBraille, développé par Mattheiss et al.[Mattheiss et al., 2015], pose les points Braille 1, 4, 5 et 6 dans les coins de l'écran et les points 2 et 3 sur les longs bords de l'écran, et oblige les utilisateurs à faire glisser un doigt le long des bords de l'écran pour activer les points Braille. L'utilisateur doit déplacer son doigt pour tracer une ligne continue sur les points Braille représentant la lettre souhaitée. L'auteur a également développé une autre version de EdgeBraille qui permet aux utilisateurs d'entrer huit points pour le Braille. Les deux points supplémentaires permettent aux utilisateurs d'effectuer d'autres fonctions telles que la suppression et la recherche. Ce clavier présente plusieurs avantages, y compris un retour vibro-tactile et sonore pour informer les utilisateurs sur l'activation et la désactivation des points Braille. En outre, il utilise VoiceOver pour lire le texte écrit. La principale force d'EdgeBraille est l'utilisation des bords et des coins de l'écran comme emplacements des points Braille, ce qui rend les points faciles à trouver pour un déficient visuel. En revanche, EdgeBraille n'est pas très précis et peut poser des problèmes du fait de son mécanisme de dessin : les points Braille sont séparés, obligeant les utilisateurs à tracer une ligne pour les relier. Cela peut tromper l'utilisateur, ajoutant à cela la nécessité de mémoriser d'autres formes de Braille. Il est également lent et nécessite un seuil de 75 millisecondes pour relever le doigt après chaque point activé. Ceci rend alors la saisie plus lente.

Enfin, on peut également citer la technique IFD (Input Finger Direction), une technique développée par Azenkot[Azenkot et al., 2012], appliquée au système de saisie Perkinput, qui est similaire à la saisie sur une machine à écrire Braille Perkins 11. À nouveau, une connaissance du Braille permet d'optimiser la vitesse de saisie. Il demeure cependant quelques inconvénients : l'interaction en tant que telle n'est pas faite pour une utilisation sur un seul dispositif, mais bien en ayant un dispositif pour chaque main.

FIGURE 11 – La technique IFD : la main gauche correspond aux points Braille 1,2,3 tandis que la main droite accède aux points 4,5,6.

3 Synthèse

Comme nous l'avons vu précédemment, on peut distinguer deux principales catégories de systèmes de saisie de texte sur dispositif mobile conçus pour faciliter la saisie de texte par des déficients visuels : 1) les systèmes spécifiques basés sur l'utilisation du codage des caractères en Braille et, 2) l'adaptation de système de saisie de texte grand public.

3.1 Problème de la saisie Braille

La première solution consiste à proposer un clavier spécifiquement adapté pour les déficients visuels. Ces systèmes souvent basés sur un alphabet particulier ou une grammaire de gestes représentant chaque caractère de l'alphabet Braille permettent de réduire le nombre de touches présentes à l'écran et donc facilitent leur localisation par les déficients visuels.

Cependant, le fait de proposer un nouvel alphabet ou une grammaire de gestes nécessite un temps d'apprentissage qui est souvent rédhibitoire pour la plupart des utilisateurs. En effet, un utilisateur est au départ moins performant avec un nouveau système qu'avec son système habituel, même si, en théorie, le nouveau système doit permettre à l'utilisateur d'avoir de meilleures performances sur le long terme. Ceci est dû au manque de repères, à un manque de connaissance de l'alphabet et cela demande d'apprendre à utiliser ce nouveau système. Quel que soit l'utilisateur, le temps

nécessaire pour atteindre des performances au moins égales à celles de son clavier habituel est généralement conséquent [MacKenzie and Zhang, 1999b]. L'utilisateur ne voit alors pas l'intérêt de changer de système et, par facilité, préfère garder son système habituel. C'est pourquoi aujourd'hui, bien qu'il soit reconnu que le clavier AZERTY n'est pas la meilleure disposition de caractères surtout dans le cas d'une saisie à un ou deux doigt(s), ce clavier reste le plus utilisé du fait de son utilisation dès le plus jeune âge par la plupart des utilisateurs d'ordinateur.

D'autre part, selon la National Federation of the Blind, aux États-Unis, seul environ 10% des utilisateurs de dispositifs mobiles utilisent le Braille [3]. En effet, depuis le milieu du XXe siècle, l'utilisation du Braille est en baisse constante. À son apogée dans les années 1950, plus de la moitié des enfants en situation de déficience visuelle apprenaient le Braille. Aujourd'hui, ce nombre est inférieur à 10% [Chaudhuri,]. Le taux d'analphabétisme Braille parmi les personnes ayant une déficience visuelle a explosé au cours des dernières décennies. Bien qu'il soit possible d'identifier les grands facteurs de cette tendance, il ne semble pas possible de l'arrêter.

Un clavier basé sur le Braille est alors considéré pour la majorité des utilisateurs non-voyants comme un système nécessitant un gros effort d'apprentissage, d'une part par ce nouvel alphabet et d'autre part à cause de la disposition et de l'interaction proposée.

Par conséquent, il ne nous semble pas pertinent de proposer un système de saisie basé sur ce principe, car quelle que soit son efficacité, celui-ci ne serait pas utilisé par la majorité des utilisateurs non-voyants.

3.2 Problème du retour visuel et de « *l'exploration douloureuse* »

L'autre solution consiste donc à adapter un clavier logiciel grand public en proposant un retour non visuel de l'information.

Sur des claviers logiciels, l'interaction diffère grandement d'un clavier physique :

3. "How many children in America are not taught Braille ?" : `http://www.nfb.org/nfb/Braille_Initiative.asp`

il n'y a plus d'information que l'utilisateur peut sentir au moyen de ses doigts, manipuler ou appuyer. En comparaison, l'interaction physique est riche : les touches résistent, puis s'enclenchent, puis s'enfoncent, informant alors à l'utilisateur que sa saisie est terminée. Il est très difficile de porter une telle expérience sur un écran tactile ; et à l'heure actuelle, aucune solution sur mobile n'en est capable. De ce fait, cela pousse à l'utilisation de canaux sensoriels annexes comme les canaux auditifs ou vibrotactiles.

Cependant, le problème est qu'il est très difficile de fournir la même quantité d'information que le canal visuel peut transmettre en se basant uniquement sur un retour audio [Oliveira et al., 2011]. Du fait de cette connaissance et de ces informations parcellaires, l'utilisateur a énormément de mal à localiser rapidement les différentes lettres ou touches qui l'intéressent sur le clavier.

D'après l'étude de Nicolau[Nicolau et al., 2015], la plupart des utilisateurs peuvent atteindre avec un clavier logiciel augmenté d'une synthèse de parole une vitesse moyenne de 4 MPM au bout de huit semaines d'utilisation, avec un taux d'erreurs de 4.7%. Malgré une performance toujours plus importante, l'apprentissage est long (0.3 MPM par semaine environ). La majorité des utilisateurs semble avoir une connaissance du modèle spatial du clavier de plus en plus importante : les frappes sur l'écran approchent les touches qu'ils souhaitent atteindre, les mouvements sont plus efficaces en terme de temps, tandis que les re-saisies sont plus faibles (ce qui diminue le nombre de pauses nécessaire pour un retour audio de la saisie). Au niveau des erreurs, les plus nombreuses sont les substitutions ; mais contrairement aux sujets voyants, les points de frappe ne sont pas ouvertement déplacés selon une même déviation pour le même utilisateur. En effet, ils sont plus souvent dispersés autour des bords des touches. La grande majorité (98,4%) des erreurs des participants sont corrigées, ce qui occupe environ 13% de leur temps total de saisie, et n'est utile que dans les deux tiers des cas car les utilisateurs ont tendance à effacer aussi des caractères correctement saisis.

Pour eux, cela implique que la correction doit être plus simple, plus efficace et plus efficiente ; et qu'elle puisse rendre les fonctionnalités avancées plus accessibles : de nombreux sujets en effet n'utilisent pas les fonctionnalités d'autocorrection ou d'autocomplétion. Il faut également que la saisie et le retour audio soient synchronisés

(64% des erreurs de substitution viendraient d'une mauvaise correspondance entre le retour audio et la frappe).

Chapitre II

DUCK : un clavier déductif

A partir des constats obtenus du chapitre précédent, nous avons choisi d'axer nos recherches autour d'un clavier logiciel classique avec des interactions adaptées à un usage par des personnes non-voyantes. Notre premier objectif était de proposer un clavier pour lequel l'utilisateur n'a que très peu d'apprentissage à effectuer avant d'en tirer pleinement profit.

Notre système s'appuie donc sur les bases d'un clavier logiciel standard, utilisant une disposition AZERTY, bien connu des personnes non-voyantes, notamment avec l'utilisation de systèmes de suppléance du type VoiceOver ou Talkback. Très peu d'apprentissage est donc nécessaire pour maîtriser la disposition spatiale des caractères de notre clavier, ainsi que les interactions utilisées.

Nos objectifs sont, d'une part, d'améliorer les performances de saisie d'un utilisateur avec ce type de clavier ; nous cherchons à la fois à augmenter la vitesse de saisie de texte tout en réduisant le taux d'erreurs. D'autre part, de permettre une prise en main très rapide par les utilisateurs novices.

Nous commencerons par décrire le fonctionnement d'un clavier de type VoiceOver pour en montrer les points forts et les faiblesses. Nous présenterons ensuite le fonctionnement de notre clavier, avec une explication détaillée de l'algorithme permettant de déduire les mots saisis par l'utilisateur. Enfin, nous présenterons la première étude que nous avons réalisée sur ce système afin de montrer la pertinence, mais aussi les

limites de notre approche.

1 Fonctionnement d'un clavier logiciel oralisé

Pour expliquer le fonctionnement d'un clavier logiciel oralisé et utilisé par un utilisateur déficient visuel, nous détaillerons le fonctionnement des deux principales caractéristiques présentes dans la plupart des dispositifs (AOSP Keyboard sous Android, ou DK sous IOS) : la saisie d'un caractère et la gestion des erreurs.

1.1 Phase de saisie

Sur un clavier logiciel classique où l'utilisateur voit le clavier et donc la disposition spatiale des caractères à l'écran, l'utilisateur clique, c'est-à-dire appuie sur la touche représentant le caractère qu'il souhaite saisir et la relâche immédiatement.

Dans le cas d'une saisie non visuelle, une étape supplémentaire est nécessaire entre la pression et le relâchement du doigt sur l'écran. En effet, même si l'utilisateur a une bonne connaissance du clavier, il va attendre d'avoir un retour audio lui confirmant le caractère se trouvant sous son doigt avant de relever son doigt de l'écran. Si le caractère est bien celui qu'il souhaite, il peut alors immédiatement relever son doigt. Dans le cas inverse, tant que l'utilisateur n'a pas relevé son doigt, il peut le déplacer sur l'écran jusqu'à le positionner sur le caractère souhaité. A chaque fois que le doigt passe sur un nouveau caractère, ce dernier est alors oralisé. Ainsi l'utilisateur a un retour sur sa position à l'écran. Comme dans le premier cas, lorsque l'utilisateur est positionné sur le bon caractère, il peut relever son doigt pour valider sa saisie. Cette étape supplémentaire est appelée phase de recherche car c'est celle qui permet à l'utilisateur de trouver le caractère souhaité.

Cette phase de recherche présente deux inconvénients majeurs. D'une part, l'utilisateur perd forcément du temps par rapport à un utilisateur voyant : même dans le cas où il est sur le bon caractère, l'utilisateur marque un temps d'arrêt pour obtenir le retour audio lui signifiant le caractère sur lequel il est positionné. Ce temps d'arrêt est pénalisant par rapport à un clic où l'utilisateur appuie et relâche immédiatement.

D'autre part, dans le cas où l'utilisateur n'est pas directement situé sur le bon caractère, il doit alors déplacer son doigt à l'écran pour trouver le caractère souhaité, ce qui peut être plus ou moins long selon sa connaissance de la disposition spatiale des caractères sur le clavier.

Le second inconvénient de cette phase de recherche est la précision de la saisie effectuée. En effet, même si le doigt se trouve sur la touche souhaitée, aucune information n'est donnée à l'utilisateur sur la position précise de son doigt sur la touche. Ainsi, il peut être sur le bord de la touche, à la frontière avec une autre touche et, au moment du relâchement du doigt, si celui-ci effectue un léger déplacement en relevant son doigt, l'utilisateur risque de sélectionner la touche voisine et le caractère validé sera erroné. Enfin, la validation s'effectuant au moment où le doigt n'est plus en contact avec l'écran, une validation intempestive peut se produire à tout moment si l'utilisateur, par mégarde, décolle même légèrement son doigt de l'écran lors de la phase de recherche.

Dans le cas de la saisie d'un mot, il est nécessaire de réitérer le processus de saisie d'un caractère tel qu'énoncé ci-dessus pour toutes les lettres du mot. La succession de ces phases de recherche peut s'avérer fatigante pour l'utilisateur et provoquer au cours de la saisie une baisse de la concentration. A force de se focaliser sur la recherche du caractère en cours, l'utilisateur peut par exemple oublier le mot qu'il est en train de saisir ou faire des erreurs d'inattention causées par la trop forte attention portée au caractère qu'il est en train de rechercher.

Cette phase de recherche apparaît donc comme étant fortement pénalisante pour l'utilisateur. D'une part par le temps qu'elle nécessite, et d'autre part par le flot d'informations vocales qu'elle génère et qui peut pénaliser à long terme l'attention de l'utilisateur. Enfin, malgré cette phase qui doit normalement aider l'utilisateur à effectuer sa saisie avec précision, on constate que plusieurs erreurs, de différentes natures, peuvent quand même se produire.

1.2 Gestion des erreurs

Pour pallier ces erreurs, les claviers logiciels existants proposent trois mécanismes de correction.

1.2.1 Correction manuelle

Cette première correction est classique ; elle nécessite l'utilisation d'un curseur pour se positionner à l'endroit de l'erreur. Lorsque l'utilisateur souhaite corriger sa saisie, il peut appuyer sur l'endroit précis de son erreur, et corriger ses erreurs en insérant ou supprimant des caractères. Cette méthode est extrêmement difficile à utiliser pour un utilisateur en situation de déficience visuelle. En effet, les non-voyants n'utilisent pas de système de pointage. Par conséquent, ils ne vont pas pouvoir positionner directement le curseur à l'endroit de la correction à effectuer. Pour y parvenir, ils doivent utiliser les touches du clavier comme la touche retour arrière ou les touches directionnelles pour déplacer avec précision le curseur jusqu'à l'erreur. L'utilisation de ces touches peut s'avérer très coûteuse en temps si la correction à effectuer est située loin du curseur actuel. De plus, une fois la correction effectuée, il faudra repositionner le curseur au bon endroit pour pouvoir continuer la saisie.

1.2.2 Correction automatique

Une autre méthode possible est la correction automatique une fois le mot fini. Pour cela, le système analyse la saisie de l'utilisateur et la corrige si nécessaire en remplaçant le mot tapé par le mot le plus proche dans le dictionnaire de correction. Ce mot est brièvement affiché à l'utilisateur lors de sa saisie (voir figure 12).

Cette méthode est jugée difficile, et souvent source d'erreurs de saisie par de nombreux utilisateurs voyants [Madison, 2012]. La correction automatique est utilisable, mais elle pose souvent des difficultés car la correction est effectuée de manière implicite, sans demander de confirmation à l'utilisateur. Si la correction effectuée ne convient pas à l'utilisateur, il doit alors lui-même revenir dessus pour la corriger et perd à nouveau du temps.

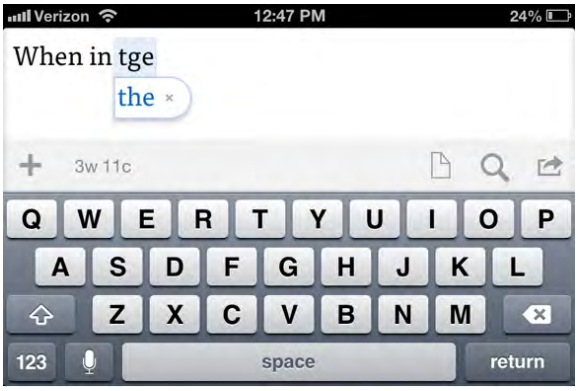

FIGURE 12 – Utilisation de la correction automatique

Une autre solution consiste à proposer à l'utilisateur le mot, ou une liste de mots, par lequel il souhaiterait remplacer sa saisie. Cette phase de validation peut soit être proposée automatiquement à l'utilisateur en fin de mot et donc celui-ci est obligé de la prendre en compte ; soit être proposée sur un composant à côté du clavier et l'utilisateur peut consulter les propositions s'il le souhaite. Dans tous les cas, cela demande une interaction et un retour vocal supplémentaires pour valider ou non la correction proposée, l'utilisateur est alors freiné dans sa saisie pour chaque mot jugé incorrect par le système.

Cette solution permet ainsi de réduire les erreurs en proposant à l'utilisateur de les corriger à chaque fin de mot. En revanche, cette étape supplémentaire vient encore réduire la vitesse de saisie de texte ; car elle oblige l'utilisateur à effectuer des actions supplémentaires, alors qu'il s'était déjà appliqué à saisir chaque caractère de son mot.

1.2.3 Système de prédiction

Pour permettre à l'utilisateur d'anticiper la saisie complète de son mot, les claviers logiciels sur mobile présentent généralement une liste de prédiction à l'utilisateur. Cette liste est souvent affichée juste au-dessus du clavier (voir figure 13). L'utilisateur peut alors sélectionner le mot désiré, dès que celui-ci apparaît dans la liste, et ainsi éviter de saisir l'ensemble de la chaîne de caractères. Les systèmes de prédiction ont

aussi l'avantage de proposer des mots correctement orthographiés : c'est-à-dire que même si l'utilisateur a réalisé une faute de frappe, le système de prédiction peut lui proposer des mots fréquents proches de ce qu'il a saisi. L'utilisateur peut donc être un peu moins précis dans sa saisie et la corriger rapidement grâce à cette liste de mots.

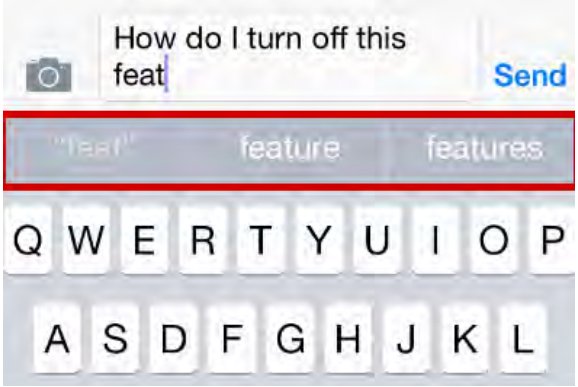

FIGURE 13 – Utilisation de listes de prédiction

Cependant, cette liste représente un focus d'attention supplémentaire, et détourne le regard de l'utilisateur de sa saisie. Sa vitesse de saisie peut alors s'en retrouver diminuée si celui-ci ne trouve pas le mot qu'il recherche dans la liste. Pour des utilisateurs voyants, l'affichage de la liste de mots n'est pas trop pénalisante car ils peuvent y jeter un regard en continuant de saisir leur mot sans trop ralentir leur vitesse de saisie.

Dans le cas des utilisateurs non-voyants, le problème est différent : pour que l'utilisateur prenne connaissance des mots qui lui sont proposés, il faut que ces derniers soient oralisés. Or, la liste de mots change après chaque caractère saisi. Par conséquent, oraliser chaque mot de la liste serait beaucoup trop perturbant au cours de la saisie de l'utilisateur. Le système annonce déjà par synthèse vocale les caractères survolés par l'utilisateur au moment de sa recherche d'un caractère sur le clavier. Les deux retours auditifs ne pourraient donc pas être faits en parallèle : il n'est donc pas possible de présenter la liste de mots pendant la saisie d'un mot.

Une autre solution consisterait à ce que l'utilisateur parcoure cette liste quand

il le souhaite. Cela nécessiterait que l'utilisateur interrompe sa saisie pour parcourir la liste avec le doigt et prenne connaissance des mots qui s'y trouvent. Si le mot recherché ne se trouve pas dans la liste, l'utilisateur retournera alors continuer sa saisie sur le clavier. Faire plusieurs allers/retours entre la liste et le clavier pendant la saisie d'un mot hacherait trop sa saisie et la vitesse de saisie en serait grandement réduite.

Cette solution à base de liste de mots au cours de la saisie n'est donc pas envisageable dans le cas d'un clavier optimisé pour des déficients visuels.

En résumé, bien que la correction soit souvent nécessaire pour corriger les fautes de frappes de l'utilisateur, cette phase de correction ou prédiction est coûteuse en temps pour les utilisateurs, et plus particulièrement pour les utilisateurs non-voyants.

2 Clavier DUCK

2.1 Choix de conception

Les principaux constats que nous pouvons faire de l'étude du système de saisie de texte précédemment décrit sont : d'une part, une phase de recherche de chaque caractère trop coûteuse en temps pour l'utilisateur, et ce malgré une bonne connaissance de la disposition spatiale des caractères sur le clavier [Merlin and Raynal, 2012] ; d'autre part, un nombre d'erreurs de saisie assez conséquent. Au regard de ces deux éléments, il paraît nécessaire de faire appel à un système basé sur des connaissances linguistiques afin de réduire le nombre d'erreurs. Enfin, il semble important de réduire l'utilisation du retour vocal pour éviter de submerger l'utilisateur d'un trop grand nombre de messages audio.

Pour cela, il nous semblait essentiel de conserver une disposition des caractères déjà connue de la majorité des utilisateurs déficients visuels. De la même manière que VoiceOver, nous avons donc conservé celle du clavier AZERTY qui est le clavier physique qu'utilise la majorité des personnes en France. Nous partons de l'hypothèse que les utilisateurs familiers du clavier AZERTY ont une bonne connaissance de la disposition spatiale des caractères, qui leur permet de situer tous les caractères, au

moins de manière approximative.

Pour éviter la phase de recherche de chaque caractère, nous partons d'un paradigme de saisie différent de celui utilisé par la majorité des claviers : nous ne considérons pas la saisie comme une suite de caractères entrés consécutivement, mais comme la saisie de mots. Ainsi, nous souhaitons laisser la possibilité à l'utilisateur de pouvoir taper approximativement sur le clavier à l'endroit où il estime que le caractère souhaité se situe. Avec l'ensemble des frappes approximatives pour un mot, nous pensons qu'un système de déduction est en mesure de retrouver le mot que l'utilisateur a souhaité saisir.

Cette approche présente un double avantage : d'une part, l'intérêt de réaliser une saisie approximative est de supprimer complètement la phase de recherche pour ces caractères. Les utilisateurs sélectionnent alors les caractères de la même manière que les utilisateurs voyants, c'est-à-dire par un clic sur l'écran. D'autre part, le système de déduction est basé sur un dictionnaire. Par conséquent, les mots proposés sont correctement orthographiés, ce qui devrait ainsi permettre à l'utilisateur de réduire le nombre d'erreurs.

Enfin, les frappes étant approximatives, un retour vocal est inutile. Annoncer à l'utilisateur un caractère erroné risque de le perturber. Le retour vocal principal se fait donc au niveau du mot et non plus au niveau du caractère. En revanche pour conserver un retour sur chaque action de l'utilisateur, nous proposons d'émettre un son à chaque frappe de l'utilisateur.

En résumé, lors de l'utilisation de notre clavier, nous ne considérons plus la saisie comme la frappe d'une suite de caractères indépendants, mais comme la saisie de mots. Notre objectif est de gagner du temps sur la saisie de la suite de caractères qui forme le mot en permettant une saisie approximative des caractères. Nous pensons qu'en utilisant un système de déduction à la suite de cette séquence de frappes approximatives, le système permettra de proposer le mot correctement orthographié à l'utilisateur.

2.2 Choix des interactions

Le modèle de saisie que nous souhaitons utiliser (saisie de mots), est déjà utilisé dans le cadre de la saisie de texte sur dispositif mobile. Le système de saisie de texte utilisant ce principe est le clavier SHARK [Kristensson and Zhai, 2004].

Pour saisir un mot, l'utilisateur effectue un tracé, avec le doigt restant appuyé sur l'écran, en parcourant successivement l'ensemble des caractères composant le mot à saisir. Un algorithme déduit alors les mots possibles à partir du tracé effectué par l'utilisateur. Deux types d'algorithmes sont possibles pour déduire le mot à partir du tracé : soit l'algorithme se base sur l'ensemble des caractères sur lesquels l'utilisateur est passé en effectuant son tracé [Mankoff and Abowd, 1998] ; soit l'algorithme se base sur la forme du tracé pour en déduire le mot souhaité [Zhai and Kristensson, 2007].

Dans notre cas, il n'était pas possible de se baser sur une interaction par tracé de l'utilisateur. Comme relevé par Kamel et Landay [Kamel and Landay, 2000], lorsqu'un voyant dessine ou suit un contour, il ajuste constamment son tracé en fonction de son parcours. Ce processus est essentiel pour capturer un modèle depuis le monde réel et le représenter sur un autre médium. Pour saisir un mot au moyen d'un tracé, l'utilisateur va anticiper son tracé en visualisant la position du caractère suivant sur le clavier. Les traits du tracé sont souvent rectilignes car l'utilisateur programme son geste à l'avance. Quand il quitte un caractère pour un autre, il sait déjà où se situe le suivant et a programmé son geste pour que celui-ci soit le plus direct possible.

Ce processus ne peut pas être utilisé par une personne non-voyante. Même si l'utilisateur non-voyant connaît bien son clavier, deux cas de figure sont envisageables : soit il n'a pas de retour vocal au cours de la saisie du mot (comme nous le proposons pour le système DUCK), et celui-ci va pointer approximativement les lettres du mot. De ce fait, il ne passera pas forcément sur les bons caractères. Soit, le système propose un retour vocal à l'utilisateur au cours de son tracé, mais dans ce cas-là, l'utilisateur ajustera son tracé entre chaque caractère en fonction du retour obtenu. Le tracé entre deux caractères ne sera alors pas aussi rectiligne et il deviendra alors

plus difficile à interpréter.

Dans les deux cas, en étant approximatif sur chacun des caractères saisis, ou en ayant des trajectoires réajustées en permanence, les deux algorithmes proposés pour déduire un mot à partir d'un tracé ne pourront pas fonctionner correctement. Avec le premier algorithme, si la suite des caractères survolés ne contient pas les bons caractères, l'algorithme ne pourra pas déduire le bon mot. Avec une suite de caractères trop approximative, l'algorithme proposera certainement d'autres mots se situant dans la séquence de caractères survolés. Dans le cas du second type d'algorithme, les approximations vont donner un tracé différent, avec des longueurs de segments et des angles différents, ce qui rendra difficile la reconnaissance du bon mot.

C'est pourquoi nous avons choisi de nous baser sur un ensemble de frappes de l'utilisateur, correspondant à chaque caractère du mot, plutôt qu'à un tracé de l'utilisateur. Ainsi, le système de déduction fonctionne à partir d'un nombre de points correspondant au nombre de caractères composant le mot. Toutes les informations (traits du tracé) effectuées entre deux caractères du mot sont donc supprimées. Ainsi, en réduisant au maximum les informations nécessaires pour déduire le mot, nous espérons limiter les erreurs d'approximations qui pourraient se cumuler lors du tracé.

2.3 Fonctionnement

Sur notre clavier, la saisie d'un mot par un utilisateur se fait de la manière suivante : l'utilisateur doit commencer par entrer précisément le premier caractère du mot. Cela se fait de la même manière que pour le clavier VoiceOver : l'utilisateur pose son doigt sur l'écran à l'endroit où il le souhaite, le système lui indique alors par synthèse vocale le caractère s'y trouvant. L'utilisateur peut déplacer son doigt sur l'écran pour parcourir le clavier. A chaque fois que celui-ci est au-dessus d'une nouvelle touche, la synthèse vocale lui indique le nouveau caractère survolé. Lorsque le doigt est positionné sur le caractère souhaité, l'utilisateur n'a plus qu'à relever le doigt de l'écran pour valider ce caractère.

Une fois ce premier caractère saisi, la saisie se fait de manière approximative pour le reste des caractères qui composent le mot souhaité : l'utilisateur va simplement

taper sur l'écran à l'endroit où il pense que se trouve le caractère suivant qu'il veut saisir. Une fois tous les caractères du mot saisis de cette manière, l'utilisateur valide sa saisie par une double frappe à deux doigts sur l'écran. Le système de déduction peut alors, à partir des actions de l'utilisateur, proposer la liste des mots les plus proches des frappes réalisées. La liste de mots proposée à l'utilisateur remplace alors le clavier, et l'utilisateur choisit parmi ces mots celui qui correspond.

Dans le cas où un seul mot est proposé par le système de déduction, ce mot est validé automatiquement et l'utilisateur est averti par un retour vocal lui indiquant le mot qui vient d'être inséré. Enfin, dans le cas où aucun mot ne peut être proposé par le système de déduction, l'utilisateur est averti de son erreur par un retour sonore, et l'utilisateur doit alors recommencer sa saisie.

Une telle approche permet de rester sur des interactions très simples à base de pression et relâchement du doigt sur l'écran que les utilisateurs non-voyants ont l'habitude d'utiliser sur Smartphone. Elle permet aussi de limiter l'utilisation des gestes, même simples (glisser à droite, glisser en haut, etc.), qui peuvent être difficiles à réaliser pour un déficient visuel ou source de multiples erreurs.

2.4 Implémentation

2.4.1 Agencement des caractères

Le système DUCK peut s'appliquer à n'importe quelle disposition de caractères. Pour pouvoir prédire les mots, le système de déduction a simplement besoin de connaître la position de chaque caractère à l'écran pour déduire le mot souhaité par l'utilisateur en fonction de ses frappes.

Pour notre étude, nous avons choisi d'utiliser un clavier qui occupe l'intégralité de la surface de l'écran. Afin de pouvoir s'adapter à toute taille d'écran, toutes les positions sont exprimées de manière relative par rapport à la taille de l'écran. Ainsi, chaque coordonnée est comprise entre 0 et 1, pour représenter sa position relative, en pourcentage de la taille de l'écran. Le coin supérieur gauche de l'écran représente l'origine du clavier et a comme coordonnées (0, 0). Chaque caractère c_i du clavier, avec c_i appartenant à l'alphabet A, est alors défini par une position (x_i, y_i) à l'écran

qui correspond au centre de la touche t_i à laquelle il est lié.

2.4.2 Dictionnaire utilisé

Pour permettre la déduction du mot tapé en fonction des frappes de l'utilisateur, nous utilisons un système de déduction. Tous les mots sont indexés dans un dictionnaire sous forme de couples contenant leur première lettre et leur longueur. Ainsi la déduction des mots ne se fait que sur les mots qui commencent par la même lettre que la première lettre saisie par l'utilisateur et qui ont la même longueur que le nombre de frappes effectuées par l'utilisateur.

2.4.3 Algorithme de déduction

La première étape consiste à présélectionner les mots du dictionnaire qui correspondent aux deux critères précédemment décrits : l'algorithme ne garde que les mots commençant par la même première lettre que la première lettre saisie par l'utilisateur et qui ont la même longueur que le nombre de frappes effectuées par l'utilisateur.

Une fois ce premier filtre appliqué, notre algorithme fonctionne sur un calcul de la distance séparant chaque mot w du sous-ensemble E correspondant à la séquence de frappes utilisateur. En considérant que l'utilisateur a tapé une séquence de N frappes, $h = [h_i]$, où $h_i = (x'_i, y'_i)$ sont les coordonnées de la $i^{\text{ème}}$ frappe, et que chaque mot w de E peut être décrit comme une séquence de caractères $w = [c_i]$, la distance séparant le mot w de la séquence de frappes h est obtenue par :

$$D(h, w) = \sum_{i=0}^{N-1} \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}$$

Ainsi, nous obtenons pour chaque mot du dictionnaire une distance le séparant de la séquence de frappes effectuées par l'utilisateur. Le système de déduction retourne alors les 4 mots qui ont la plus faible distance.

2.5 Hypothèses

Nous formulons deux principales hypothèses : D'une part, en demandant à l'utilisateur une saisie plus approximative, ne nécessitant plus de phase de recherche précise du caractère à saisir, la vitesse de saisie de l'utilisateur devrait être augmentée, et ce malgré l'utilisation obligatoire d'un système de déduction à la fin de chaque mot.

D'autre part, la saisie au niveau du mot et non du caractère, couplée à l'utilisation obligatoire du système de déduction devrait permettre de réduire le nombre d'erreurs de saisie que l'utilisateur peut commettre avec les claviers actuels.

Chapitre III

Première étude expérimentale

Cette première étude a pour but de vérifier la pertinence de notre système DUCK, ainsi que les hypothèses que nous avions formulées au moment de sa conception. Cette étude était assez courte, c'est à dire, sur une seule session pour chaque utilisateur. Elle a pour but de détecter d'éventuels problèmes de conception, ainsi que d'analyser les points sur lesquels notre système pourrait être optimisé.

1 Protocole

1.1 Participants

Nous avons recruté douze participants, six hommes et six femmes (âge moyen : 32,4 ans, écart-type : 16,2 années). Dix d'entre eux étaient considérés comme aveugles sans aucune perception. Deux utilisateurs avaient une perception de luminosité. Aucun d'entre eux n'avait de capacités visuelles suffisantes pour percevoir la forme générale du clavier virtuel. Tous les participants ont déclaré avoir une bonne connaissance de la disposition de caractères AZERTY : huit utilisateurs en avaient une utilisation quotidienne personnelle ; tandis que quatre donnaient des cours de dactylographie, bénéficiant de ce fait d'une connaissance dite experte.

1.2 Dispositif

L'expérience a été menée sur un Samsung Galaxy S III, en utilisant une version stock d'Android 4.1, un quadruple cœur de 1,4 GHz Cortex-A9 pour CPU et 1 Go de RAM. L'écran tactile a une taille de 59,8 × 106,2 mm avec une résolution de 720 × 1280 pixels. L'ensemble du dispositif d'expérimentation (c'est-à-dire les claviers à tester ainsi que l'application permettant de présenter la tâche à l'utilisateur) est codé en Java pour une utilisation directe sur le smartphone. La synthèse vocale est proposée par le moteur IVONA, que nous avons sélectionné pour sa clarté et sa prosodie proche d'une voix naturelle.

Nous avons utilisé notre système DUCK sur un clavier logiciel minimaliste sur lequel nous avons uniquement utilisé les 26 lettres de l'alphabet, selon la disposition AZERTY. Cet agencement que nous avons baptisé « *AZERTY réduit* » comprend deux rangées de dix touches et une rangée de six touches, n'ayant aucune possibilité de saisir un signe de ponctuation quelconque. De plus, à la différence des claviers physiques, les touches étaient alignées sur toutes les lignes : il n'y avait pas de demie touche de décalage entre les touches de la deuxième ligne et celles de la première.

Pour valider nos hypothèses, nous avons choisi de réaliser une étude comparative entre notre système DUCK et un clavier logiciel utilisant la technique d'interaction couramment utilisée avec des systèmes d'aide à la saisie présents dans les dispositifs mobiles les plus courants (VoiceOver d'Apple, TalkBack de Google). Cependant, pour nous limiter à une étude comparative sur les interactions, nous n'avons pas directement utilisé un de ces systèmes, mais nous avons choisi de reconcevoir un clavier reprenant l'interaction de ces systèmes. Nous avons nommé cette reproduction VODKA. Pour concevoir VODKA, nous avons simplement implémenté la technique « *d'exploration douloureuse* » : chaque caractère est prononcé par la synthèse vocale quand l'utilisateur glisse son doigt au-dessus, et est validé au moment où l'utilisateur lève son doigt. Comme nous nous sommes uniquement focalisés sur l'interaction, nous n'avons ajouté aucun système de correction, prédiction ou déduction existant au clavier VODKA. Enfin, afin de pouvoir comparer les deux claviers lors de notre étude, nous avons utilisé le même agencement de caractères pour les deux claviers.

Pour les deux claviers, les utilisateurs peuvent effacer la dernière lettre tapée en effectuant un geste avec deux doigts vers la gauche. De même, il est possible d'entrer un espace en faisant glisser deux doigts vers la droite. Les espaces sont toutefois automatiquement ajoutés avec DUCK dès que le mot est validé.

1.3 Procédure

La tâche mesurée consistait à taper une phrase courte après dictée par la synthèse vocale, en utilisant soit DUCK, soit VODKA. Les participants devaient taper cette phrase aussi vite que possible tout en minimisant le nombre d'erreurs.

Toutes les phrases à saisir étaient simples afin de faciliter leur mémorisation, mais également pour minimiser le nombre d'erreurs liées à une mauvaise orthographe. Chaque phrase comprenait quatre à sept mots parmi les mots les plus fréquents de la langue française, en s'assurant que les lettres de ces phrases correspondaient à la répartition de la fréquence des lettres de la langue française.

Chaque exercice débutait par une brève phase de familiarisation pour le clavier étudié. Les participants devaient apprendre à saisir des lettres, des mots courts, des mots longs et enfin des phrases. A tout moment, l'utilisateur pouvait réécouter la phrase à saisir en faisant glisser deux doigts vers le haut. Il pouvait écouter sa saisie réelle en faisant glisser deux doigts vers le bas, et effacer sa dernière entrée (mot pour DUCK, lettre pour VODKA) en faisant glisser deux doigts vers la gauche. À la fin de chaque exercice, l'utilisateur devait compléter un questionnaire SUS sur le clavier. En outre, une fois l'expérience terminée, ce dernier devait mentionner sa préférence générale et ses opinions sur les deux claviers.

1.4 Conception

Chaque participant a réalisé deux exercices : un avec chaque système (DUCK et VODKA). La moitié des utilisateurs a commencé avec une session utilisant DUCK, tandis que les autres ont commencé avec VODKA. Au sein de chaque exercice, l'utilisateur devait saisir huit phrases courtes sous la dictée. Au total, chaque participant a

saisi 16 phrases (8 phrases avec chaque système). Ces seize phrases étaient identiques pour chaque participant, mais distribuées dans un ordre pseudo-aléatoire différent, de manière à ce que chaque phrase aient été saisie autant de fois avec chaque système à la fin de l'expérience.

Nous avons enregistré les saisies effectives de douze utilisateurs différents tapant seize phrases de quatre à sept mots, soit un total de 192 phrases et 1 032 mots.

1.5 Données collectées

Pour pouvoir analyser les résultats de cette expérience, nous avons enregistré dans un fichier XML, lors de chaque session, l'ensemble des actions effectuées par l'utilisateur (pression, relâchement, et déplacements du doigt à l'écran), les instructions données par le système à l'utilisateur (consigne de l'exercice, phrases à recopier) ainsi que les listes de mots qui étaient présentées à l'utilisateur au cours de la saisie. Enfin, nous avons aussi recueilli les réponses obtenues au questionnaire SUS, ainsi que les préférences et observations données par chaque participant.

2 Résultats

Pour chaque sujet, l'expérience entière a duré entre 50 à 165 minutes, comprenant une pause de 15 minutes entre les deux sessions. Dans les sections suivantes, le niveau de signification (α) a été fixé à 0,05.

Les sections suivantes présentent les résultats quantitatifs et qualitatifs, ainsi que des statistiques déductives. Les corrections de Bonferroni ont été systématiquement appliquées pour les comparaisons multiples.

2.1 Vitesse de frappe pour les mots et les phrases

Afin de comparer les performances des utilisateurs lors de l'utilisation des deux claviers, nous avons d'abord sélectionné les phrases et les mots qui ont été tapés correctement. L'analyse sur les erreurs de frappe est détaillée dans la section suivante.

Tout d'abord, nous avons calculé la vitesse de saisie de texte par phrase. Ici, la longueur de chaque phrase (y compris les espaces entre les mots et à la fin d'une phrase) a été divisé par le temps (en secondes) nécessaire pour entrer cette phrase. Cette vitesse est donnée en caractères par seconde (CPS).

Les vitesses moyennes de saisie des phrases étaient de 0,37 cps ($\sigma = 0,12$) pour VODKA et de 0,38 cps ($\sigma = 0,15$) pour DUCK, avec une médiane de 0,33 cps pour les deux systèmes. Comme les distributions de vitesse ne sont pas normales (Shapiro-Wilk, W = 0,95, p <0,01), nous avons utilisé un test de Wilcoxon pour les comparer. Ce test a montré que les vitesses ne sont pas statistiquement différentes (W = 2,115, p = 0,819) entre les deux claviers.

Nous avons également analysé la vitesse de saisie plus finement en calculant la vitesse de saisie au niveau du mot. Cette analyse permet de supprimer le temps de pause ou de réflexion que pourrait prendre l'utilisateur entre la saisie de deux mots. Nous pensons que si l'utilisateur marque une pause, il le fera plus facilement après avoir terminé de saisir un mot plutôt qu'en plein milieu du mot. Le calcul de la vitesse de saisie d'un mot correspond à la longueur de ce mot divisée par la durée obtenue entre le moment de la saisie du premier caractère du mot et celui de la validation finale de ce mot : c'est-à-dire au moment de l'appui sur la barre espace pour VODKA, ou au moment de l'appui des deux doigts sur l'écran pour valider un mot de la liste de déduction avec DUCK. Il est important de noter ici que le temps de saisie pour DUCK comprend la phase de validation, qui consiste à écouter les différents mots de la liste et à sélectionner celui attendu.

Les vitesses moyennes de saisie de mots sont de 0,383 cps (médiane = 0,28, σ = 0,20) pour VODKA, contre 0,398 cps (médiane = 0,32, σ = 0,18) pour DUCK. Un test de Wilcoxon a montré que ces vitesses sont significativement différentes (W = 774, p = 0,048).

Afin d'analyser si la taille du mot a une influence sur la vitesse de saisie, nous avons étudié la vitesse de saisie en fonction de la longueur du mot. La figure 14 décrit la vitesse de saisie moyenne d'un mot en fonction de la longueur de ce mot. Les résultats montrent que VODKA est plus efficace que DUCK pour saisir les mots contenant peu de lettres. En revanche, plus le mot à saisir est long et plus la vitesse de

saisie avec DUCK augmente, alors qu'elle reste quasiment constante avec le système VODKA. Nous avons comparé les vitesses pour chaque longueur de mot en utilisant une série de tests de Wilcoxon avec une correction de Bonferroni. Ces tests montrent que DUCK est systématiquement plus efficace que VODKA pour les mots de plus de six caractères.

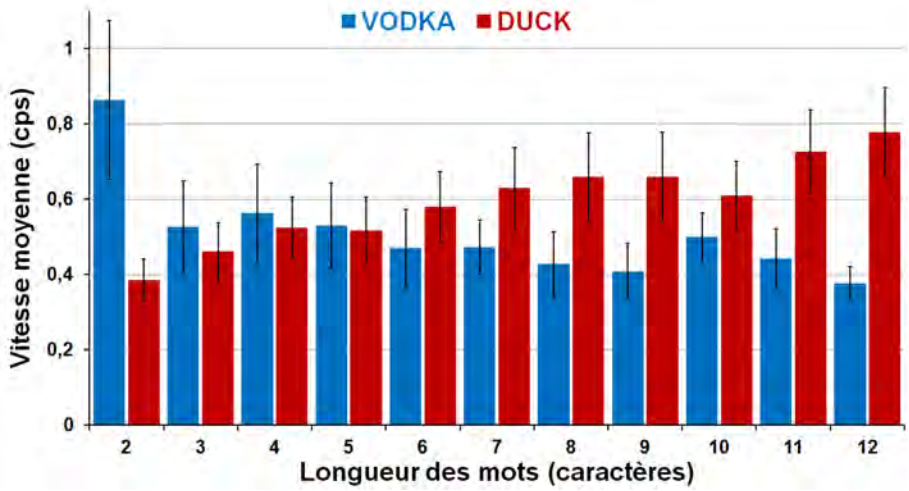

FIGURE 14 – Vitesse de frappe moyenne (en cps) en fonction de la longueur de mots pour les systèmes de saisie DUCK (en rouge) et VODKA (en bleu)

2.2 Décomposition de la saisie lors de l'utilisation de DUCK

Afin de comprendre pourquoi DUCK est plus performant pour les mots plus longs, nous avons décomposé notre analyse en fonction des étapes de saisie d'un mot avec le système DUCK. En effet, nous pouvons décomposer le temps nécessaire pour taper un mot avec DUCK en deux phases : La première phase, que nous appelons « *phase de frappes* », comprend le temps nécessaire pour trouver et sélectionner la première lettre d'un mot de N caractères, et le temps nécessaire pour effectuer les N - 1 frappes suivantes de manière approximative.

La deuxième phase, que nous appelons « *phase de validation* », comprend le temps nécessaire pour choisir le mot souhaité dans la liste de déductions proposée à

l'utilisateur. La position du mot dans la liste dépend de sa distance par rapport aux frappes de l'utilisateur. De ce fait, dans les sections suivantes, nous avons évalué le temps de saisie avec VODKA (T_V) ; et le temps de saisie avec DUCK (T_D), séparé en deux temps : le temps de frappe (T_{TAP}) et le temps de validation (T_{VAL}).

La figure 15 indique le temps nécessaire pour saisir un mot avec VODKA (T_V) et DUCK (T_D). Elle montre aussi le temps nécessaire pour la frappe (T_{TAP}) et la validation (T_{VAL}) pour DUCK. Ces différentes durées ont été calculées en fonction de la longueur de mot (2 à 12 caractères).

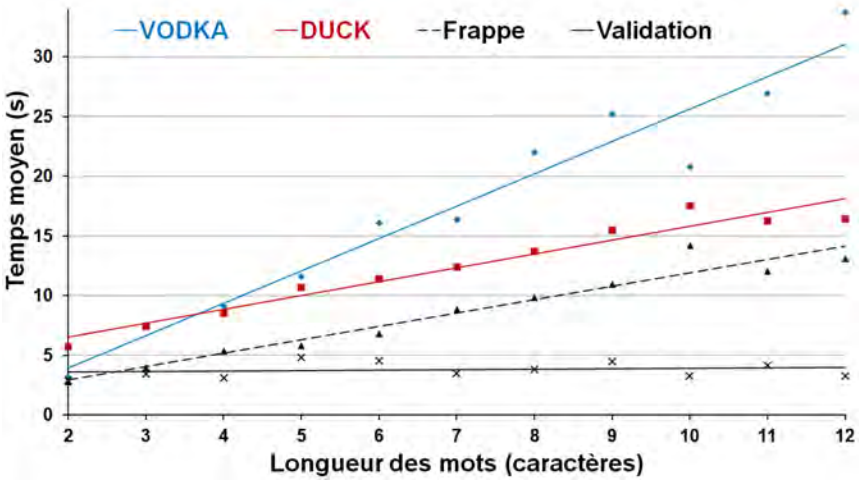

FIGURE 15 – Temps moyens (en s) en fonction des longueurs de mots. Les temps de frappe (T_{TAP}) et de validation (T_{VAL}) sont également indiqués ($T_D = T_{TAP} + T_{VAL}$)

La régression linéaire des moindres carrés, obtenue avec une équation du type $y = ax + b$, montre que le temps de saisie avec VODKA (ligne bleue) augmente avec un coefficient a de 2,7 (alors que le coefficient de corrélation R^2 est de 0,95). Le temps de saisie de DUCK (ligne rouge) est un peu moins affecté par la longueur des mots avec une pente $a = 1,2$ ($R^2 = 0,95$). Ceci montre que le temps nécessaire pour saisir avec DUCK est dépendant de la longueur du mot, mais chaque caractère en plus a un impact moindre sur le temps de saisie.

Le temps de frappe (ligne pointillée) dépend aussi de la longueur du mot. Sa progression en fonction de la longueur du mot est identique à celle de DUCK ($a = 1,1$,

avec $R^2 = 0,94$). Ceci vient confirmer notre hypothèse qu'il est plus rapide d'effectuer des frappes approximatives que de saisir précisément le caractère souhaité. Au vue de la droite de régression linéaire de VODKA et de celle de frappes avec DUCK, nous voyons que le système DUCK, grâce aux frappes approximatives, gagne en moyenne 1,5 secondes par caractère à saisir par rapport au système VODKA.

Cependant, ce gain est atténué par la phase de validation nécessaire avec le système DUCK. Ce temps de validation (ligne noire) n'est pas dépendant de la longueur du mot ($a = 0,03$), mais nécessite de la part de l'utilisateur en moyenne 3,5s.

C'est pourquoi, nous pouvons constater que le temps pour saisir des mots de moins de 4 caractères est inférieur pour le système VODKA par rapport au système DUCK. Nous pouvons constater que les droites de régression pour les temps de saisie de DUCK et VODKA se croisent au niveau des mots de 4 lettres. Ceci signifie que les mots de plus de quatre lettres sont plus rapides à saisir avec DUCK et inversement que pour les mots de 4 caractères ou moins, il est préférable de saisir avec VODKA. Cette différence est significative. Les tests de Wilcoxon avec une correction de Bonferroni donnent W = 1 402, $p = 0,02 < \alpha$ entre le temps de frappe de DUCK et le temps de saisie de VODKA.

2.3 Position du mot dans la liste

Comme mentionné précédemment, la phase de validation nécessite en moyenne 3,5 secondes, et n'est pas affectée par la longueur du mot. Toutefois, le temps nécessaire pour valider un mot dépend de sa position dans la liste (test de Wilcoxon, W = 954, p = 0,024).

Alors qu'il faut en moyenne 3,0 secondes pour sélectionner un mot situé en première position, le temps nécessaire pour valider un mot se trouvant en deuxième, troisième ou quatrième position est respectivement de 6,6 secondes, 8,2 secondes et 9,6 secondes. Cependant nous obtenons une moyenne de 3,5 secondes car dans le cas où le mot apparaît bien dans la liste, celui-ci est apparu en première position dans 83% des cas. Les mots choisis en deuxième, troisième et quatrième ne l'ont été que dans respectivement 8%, 5% et 3% des cas.

Ceci peut s'expliquer en partie par la bonne connaissance que les utilisateurs ont de leur clavier. En effet, pour les mots sélectionnés et se trouvant en première position de la liste, les frappes de l'utilisateur sur l'écran sont en moyenne à 95 px du centre de la touche associée au caractère souhaité. Pour les mots sélectionnés à d'autres positions dans la liste, la distance moyenne au centre de la touche est supérieur à 150 pixels (px) : 159 px pour les mots en deuxième position, 165 px pour les mots en troisième position et 152 px pour les mots en dernière position). Chaque touche mesurant 113 px de long et 214 px de haut, cela veut dire que les frappes de l'utilisateur sont soit sur la bonne touche, soit dans un périmètre très proche de la touche voulue.

2.4 Niveau de correction de la saisie de texte

Nous avons considéré qu'une phrase est correcte lorsque l'utilisateur n'a pas fait la moindre erreur dans la phrase. Les utilisateurs n'avaient pas de contraintes en termes de position du téléphone ou d'utilisation de doigts spécifiques. Ils pouvaient ou non corriger leurs erreurs avant de valider leur entrée.

Sur les 192 phrases saisies par les utilisateurs, 69 ont été correctement saisies avec DUCK, tandis que seulement 31 phrases ont été correctement recopiées avec VODKA. Sur les phrases comportant des erreurs, 156 additions et 62 omissions de lettres ont été commises avec VODKA, contre 218 additions et 118 omissions pour DUCK.

Pour chaque système, nous avons calculé le taux de caractères correctement saisis Tx_{cc}. Pour cela, nous nous sommes basés sur la métrique habituelle : la distance minimale interchaîne (Minimum String Distance, MSD) qui est définie par le nombre minimum de modifications de caractères nécessaires pour transformer une chaîne de caractères en une autre. Le taux de caractères correctement saisis est alors obtenu par :

$$Tx_{cc} = 1 - \frac{MSD}{|Texte|}$$

où MSD est le calcul de la distance minimale interchaîne et $|Texte|$ est la longueur du texte à saisir.

Ainsi, nous avons obtenu un taux d'éléments correctement saisis de 96% pour VODKA et 94% pour DUCK.

La plus grande précision, au niveau caractère, de VODKA par rapport à DUCK, s'explique par le fait que si l'utilisateur valide un mot incorrect avec le système de déduction de DUCK, cela peut engendrer l'ajout de plusieurs caractères erronés par rapport au mot à copier. Ainsi si le mot est complètement différent, le nombre de caractères erronés peut être équivalent au nombre de caractères du mot entré.

Pour confirmer cette hypothèse, nous avons adapté le calcul du taux de caractères correctement saisis en considérant le mot, à la place du caractère, comme élément de base. Nous avons ainsi calculé la distance minimale interchaîne en prenant le mot comme élément de base (MSD_{mots}) qui représente le nombre minimum de modifications uniques de mots nécessaire pour transformer une chaîne en une autre. Nous avons ensuite calculé le taux de mots correctement saisis Tx_{mc} par :

$$Tx_{mc} = 1 - \frac{MSD_{mots}}{|Texte|}$$

où MSD_{mots} est le calcul de la distance minimale interchaîne au niveau mot et $|Texte|$ est la longueur du texte à saisir.

Avec cette mesure au niveau du mot, nous obtenons ainsi un taux de mots correctement saisis de 92% pour DUCK contre seulement 41% lors de l'utilisation du système VODKA. Cette différence entre les deux systèmes est significative (W = 486, p <0,001)

2.5 Préférences des utilisateurs

Les scores obtenus avec les réponses des questionnaires SUS sont de 58,1 pour le clavier VODKA contre 70,5 pour le système DUCK. Ces résultats sont faibles, et montrent que les utilisateurs ne souhaiteraient pas utiliser ce clavier VODKA en l'état. Cependant, malgré ces scores, sept des douze utilisateurs préfèrent VODKA

au système DUCK.

Les utilisateurs ont expliqué leur préférence de VODKA en indiquant que certains des mots utilisés dans l'expérience étaient plus faciles à taper avec ce type de clavier. Pour certains, il est plus facile de s'appuyer sur la disposition du clavier et de l'écran physique pour taper des mots (« *Quand je veux saisir le 'p', par exemple, je dois seulement glisser mon doigt sur le coin en haut à droite jusqu'à ce que je le trouve.* »). Pour d'autres, ce clavier est plus adapté à un usage quotidien, notamment à l'utilisation de mots abrégés des SMS (« *Je ne tape pas des mots comme ça quand j'envoie des SMS, ils sont écrits en abrégé.* »). La dernière raison porte sur le sentiment de confiance et de contrôle (« *Je trouve ça effrayant de ne pas savoir ce qui a été tapé, même si le mot est correctement déduit par le système. Tout ce que j'entends, ce sont les bips lorsque je tape.* »). Avec VODKA, chaque lettre est vocalisée et il n'y a pas besoin d'une étape de validation finale. En outre, certains utilisateurs ont l'habitude de ce type de clavier, car il est similaire à VoiceOver sur iPhone (« *Je suis habitué à une saisie de cette façon : glisser et relâcher pour sélectionner une lettre. Il y a aussi un mode où on appuie deux fois sur les lettres, mais c'est plus lent.* »).

D'autre part, les utilisateurs préférant DUCK ont vraiment apprécié la vitesse de frappe. Très vite, ils ont remarqué qu'il était plus rapide que VODKA : si l'utilisateur possède une connaissance suffisante de la disposition du clavier à l'esprit, taper avec DUCK revient à taper sur un clavier normal (« *Je peux taper sur ce clavier aussi vite que je le fais sur mon PC.* »).

3 Discussions

A l'analyse des résultats, il apparaît que les claviers DUCK et VODKA se valent sur la saisie de l'ensemble des phrases. Cependant, si nous décomposons le temps de saisie pour le clavier DUCK en temps de frappe puis temps de validation, il apparaît que l'utilisateur peut effectuer la frappe des différents caractères plus rapidement avec le système DUCK qu'avec le clavier VODKA. En revanche, la saisie d'un mot

avec DUCK nécessite, en plus, un temps de validation du mot dans la liste des mots déduits par le système difficilement compressible (en moyenne 3,5 secondes). Par conséquent, pour que la saisie avec le système DUCK soit bénéfique pour l'utilisateur, il est nécessaire que le mot soit constitué de plus de 4 caractères. Pour les mots de 4 caractères ou moins, la phase de validation est trop pénalisante pour que l'utilisateur soit plus performant avec DUCK qu'avec l'interaction standard utilisée avec VODKA.

De cette première étude, il ressort donc la grande importance de la phase de validation du mot dans la liste de déduction. Si le mot est en première position, l'utilisateur met en moyenne 3 secondes pour le valider. En revanche, si le mot est plus éloigné dans la liste, le temps de validation est au-delà de 6 secondes et devient donc très pénalisant pour la saisie d'un mot avec le système DUCK.

Cela est d'autant plus pénalisant si le mot à saisir est court. En effet, plus le mot est court et moins l'utilisateur ne peut tirer de bénéfice de l'utilisation des frappes approximatives. D'autre part, nous avons vu que pour les mots de quatre caractères ou moins, même si le mot est en première position dans la liste de déduction, l'interaction classique reste plus avantageuse que l'utilisation du système DUCK.

De ces constats, nous avons donc choisi d'étudier les interactions proposées au moment de l'utilisation de la liste de déduction de manière à essayer de minimiser le temps nécessaire pour valider un mot dans la liste, et ce quelle que soit sa position dans la liste. D'autre part, les mots courts représentant une partie importante des mots saisis dans la langue française, nous avons également choisi de nous focaliser sur la saisie de ce type de mots afin de proposer à l'utilisateur une technique de saisie lui permettant d'entrer le plus rapidement possible ce type de mots.

Chapitre IV

Interaction avec la liste de prédiction

Comme nous l'avons mentionné au cours de l'étude présentée au chapitre précédent, l'utilisateur consacre en moyenne 3,5 secondes pour sélectionner le mot qu'il souhaite dans la liste de déduction qui lui est proposée. Bien que ce temps soit compensé pour les mots de plus de 4 lettres par une interaction appropriée, basée sur des frappes approximatives, il nous paraît important d'essayer de réduire ce temps de validation d'un mot avec le système DUCK, surtout pour les cas où le mot ne se trouve pas en première position dans la liste.

Dans ce chapitre, nous ne nous intéressons qu'à l'étude des interactions possibles avec la liste pour une utilisation par des personnes non-voyantes. Nous n'étudions pas le système de déduction en soi. Notre objectif est d'améliorer et de faciliter l'accès au mot dans la liste quelle que soit sa position dans celle-ci. Nos critères de réussite sont les mêmes que pour la saisie de texte : il s'agit d'une part d'augmenter la vitesse de sélection du mot dans la liste et d'autre part de réduire les erreurs de sélection de l'utilisateur afin d'éviter la validation d'un mot non désiré.

La problématique de la sélection d'un mot dans une liste est quasiment similaire à celle de la saisie d'un caractère sur le clavier. Cette tâche se décompose en trois phases :

Dans un premier temps, une phase d'**exploration**, au cours de laquelle le sujet parcourt la liste et découvre son contenu. A la différence du clavier, la liste de mots présentée à l'utilisateur est dynamique et est modifiée tout au long de la saisie de l'utilisateur. L'utilisateur ne peut donc pas apprendre la disposition des mots dans la liste, comme il peut le faire avec la disposition des caractères sur le clavier. Par conséquent, la phase d'exploration est obligatoire pour chaque sélection de mot, et elle ne pourra pas être compensée par un apprentissage au fur et à mesure de la saisie.

Les utilisateurs voyants prennent connaissance de cette liste en la parcourant des yeux. Par conséquent, ils ont immédiatement un aperçu des mots présents dans la liste ainsi que leur position à l'écran. Cette étape peut donc être assez rapide pour eux. En revanche, cette phase d'exploration peut s'avérer extrêmement problématique pour les utilisateurs non-voyants car sans ce retour visuel, il est plus difficile pour eux de connaître l'ensemble des éléments présents dans la liste ainsi que leur position. Rien n'indique à l'utilisateur s'il a pris connaissance de tous les mots et où sont positionnés les mots qu'il ne connaît pas encore.

Une fois l'exploration de la liste terminée, il s'agit de **sélectionner** le mot voulu. Pour les personnes non-voyantes, cette phase est souvent liée à la phase d'exploration car l'exploration de la liste consiste généralement à énoncer le mot que l'utilisateur est en train de pointer. En arrêtant son exploration, cela revient à sélectionner l'élément sur lequel il se trouve. Le problème de la sélection est donc d'assurer à l'utilisateur une bonne précision au moment du choix du mot et de stabiliser son choix afin d'éviter que ce soit l'élément voisin qui soit sélectionné.

Enfin l'utilisateur termine par une phase de **validation** qui consiste à confirmer sa sélection. Le risque lors de cette étape est de modifier la sélection en effectuant la validation. Il s'agit donc de trouver une interaction permettant une transition stable entre la phase de sélection et celle de validation.

Afin de répondre aux problèmes soulevés par ces 3 phases, nous avons mené trois études :

— Notre première étude concerne la disposition des éléments dans la liste. La disposition des éléments est importante pour les deux premières phases de la

sélection d'un mot. D'abord, pour que l'utilisateur ait conscience des différentes positions où il peut y avoir un élément. Une disposition facile à retenir lui permettra aussi de pointer plus rapidement un élément s'il connaît au préalable sa position dans la liste.

— Notre deuxième étude focalise sur le retour vocal proposé à l'utilisateur lors de la phase d'exploration. Il s'agit d'étudier différentes stratégies de restitution de l'information à l'utilisateur pour que celui-ci sache le plus rapidement possible si le mot souhaité est dans la liste et, si tel est le cas, quelle est sa position au sein de la liste.

— Enfin, notre troisième étude est axée sur l'interaction utilisée pour valider le mot à la fin de la sélection. Le but est de trouver la meilleure interaction pour que l'utilisateur puisse valider rapidement, et avec précision, le mot une fois celui-ci sélectionné dans la liste.

Nous détaillons donc dans la suite de ce chapitre ces trois études.

1 Présentation des éléments

La première étude que nous avons menée concerne la présentation des éléments à l'utilisateur. Étant donné que les utilisateurs malvoyants n'ont pas accès à l'information affichée à l'écran, il est nécessaire de minimiser le temps passé à parcourir l'ensemble des éléments présents dans la liste.

1.1 Dispositions proposées

Pour cela, nous avons choisi d'étudier différentes manières de disposer les éléments d'une liste. Nous distinguons trois catégories :

— La disposition **absolue** où tous les éléments sont présentés en même temps et ont une position absolue à l'écran. Chaque élément de la liste a une surface d'affichage identique à l'écran.

— La disposition **relative** où la position des éléments est relative à la position du doigt de l'utilisateur lorsque celui-ci pose son doigt sur l'écran. Avec cette

disposition, tous les éléments sont aussi accessibles en même temps sur l'écran. Ils ont tous une largeur similaire.

— La disposition **par page**. Dans ce cas, un seul élément est présenté à l'écran. L'utilisateur passe d'un élément à un autre par un geste sur l'écran.

1.1.1 Disposition absolue

La disposition absolue affiche la totalité de la liste de mots sur l'écran. L'écran est entièrement utilisé et divisé de sorte que chaque élément ait une surface d'affichage identique. Nous avons étudié trois configurations différentes :

— Une disposition **en ligne** où tous les éléments sont placés sur une ligne horizontale et occupent toute la hauteur de l'écran. La largeur de l'écran est divisée en parts égales entre chaque élément de la liste (cf. Figure 16 A).

— Une disposition **en colonne** où tous les éléments sont alignés verticalement et occupent toute la largeur de l'écran. Sur le même principe que pour la disposition en ligne, la hauteur de l'écran est divisée en parts égales entre chaque élément de la liste (cf. Figure 16 B).

— Une disposition **en grille** où les éléments sont disposés en plusieurs lignes et colonnes. Les éléments sont répartis sur l'écran de manière à occuper toute la surface d'affichage. La hauteur et la largeur de l'écran sont divisées en lignes et colonnes de mêmes tailles. Ainsi, chaque élément de la liste occupe une surface d'affichage de même taille à l'écran (cf. Figure 16 C).

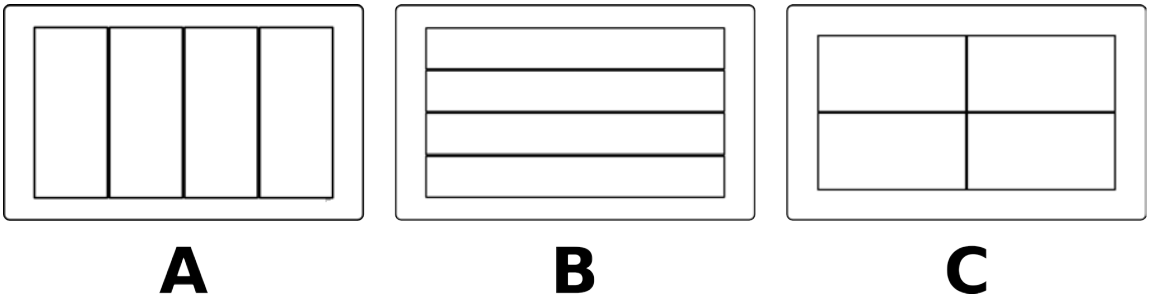

FIGURE 16 – Disposition absolue : agencement en A) ligne, B) colonne, et C) grille

Pour interagir avec la disposition absolue, nous utilisons l'interaction régulière-

ment privilégiée : l'utilisateur appuie son doigt sur l'écran, la synthèse vocale lui énonce le mot qui se situe sous son doigt. Il peut parcourir la liste en déplaçant son doigt sur l'écran. A chaque fois que le doigt survole un nouvel élément, la synthèse vocale l'énonce à l'utilisateur. La validation se fait en relâchant le doigt de l'écran, ce qui a pour conséquence de valider le dernier mot survolé par le doigt.

1.1.2 Disposition relative

La disposition relative a un fonctionnement plus dynamique. À la différence de la disposition « *absolue* », la position des éléments est liée à la position du doigt de l'utilisateur (Figure 17). Le premier élément, dont la surface d'affichage est un cercle, est centré sur la position du doigt de l'utilisateur posé à l'écran. Les éléments suivants sont représentés sous la forme d'anneaux centrés sur le premier élément (voir figure 18). Ainsi l'utilisateur peut accéder à n'importe quel élément, quelle que soit la direction qu'il décide de prendre. La largeur de l'anneau est similaire pour chaque élément. Elle correspond à la moitié de la largeur de l'écran divisée par le nombre d'éléments présents dans la liste.

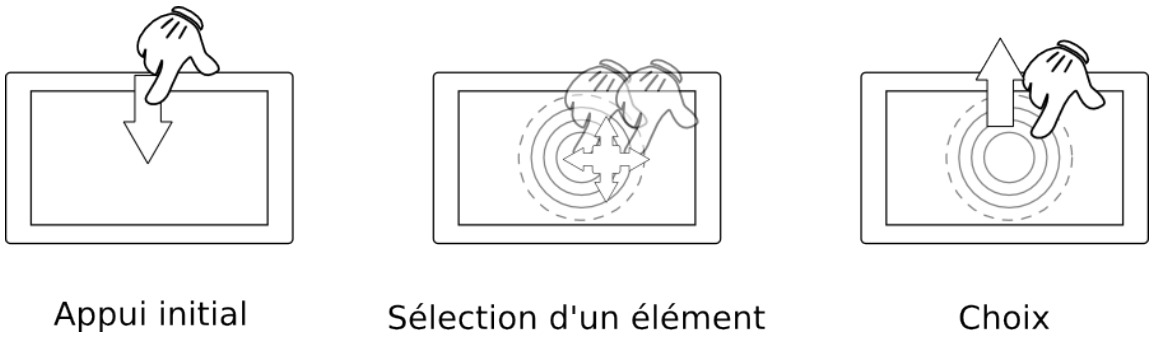

FIGURE 17 – Disposition relative

L'interaction utilisée pour manipuler ce type de disposition est la même que celle de la disposition absolue.

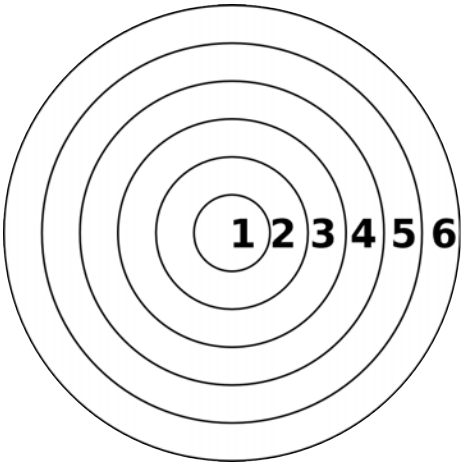

FIGURE 18 – Exemple d'une sélection relative à 6 éléments

1.1.3 Disposition par page

Dans la disposition par page, les éléments ne sont pas tous présentés sur l'écran en même temps. Un seul mot est présenté à l'écran. L'utilisateur parcourt la liste en faisant glisser son doigt vers la gauche ou la droite pour accéder respectivement à l'élément suivant ou précédent (voir figure 19). D'autre part, la disposition par page est circulaire : c'est-à-dire que lorsque l'utilisateur se trouve sur le premier élément de la liste, il peut directement accéder au dernier élément de la liste en effectuant un geste vers la droite. A chaque fois que l'utilisateur change de page, le mot présent sur la nouvelle page est énoncé par la synthèse vocale. Pour cette disposition, la validation du mot se fait au moyen d'un geste vers le bas.

Cette disposition correspond à celle qui était utilisée dans le système de déduction de DUCK lors de la première expérience présentée au chapitre précédent. Cependant, nous avons modifié les interactions initialement proposées en utilisant des gestes simples plutôt que des frappes ou double frappes sur l'écran. Nous avons fait ce choix pour permettre à l'utilisateur de revenir au mot précédent dans la liste, ce qui n'était pas le cas dans la liste de mots proposée par le système de déduction.

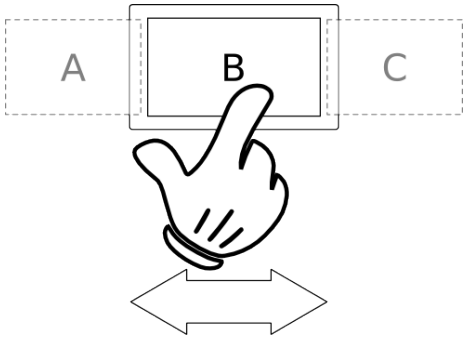

FIGURE 19 – Disposition par page

1.2 Protocole

1.2.1 Participants

12 participants (3 femmes et 9 hommes, 25 ans de moyenne d'âge) ayant une vision normale ou corrigée ont participé à notre étude. Les participants avaient les yeux bandés pour s'assurer qu'ils ne pouvaient pas voir l'écran.

1.2.2 Dispositif

Les participants ont utilisé un smartphone Samsung Galaxy SII. Le dispositif a une résolution de 306 ppm pour un écran de 136,6 mm × 70,6 mm, un processeur ARM Cortex-A9 MPCore Quad core cadencé à 1,4 GHz et un système d'exploitation Android 4.3.

Les différents énoncés et mots prononcés par la synthèse vocale étaient vocalisés à l'aide du service Google Translate. La longueur du fichier audio était ensuite minimisée pour permettre un retour audio aussi rapide que possible, minimisant ainsi tout retard. Rien n'était affiché à l'écran durant l'expérience.

1.2.3 Procédure

La tâche demandée était de trouver un mot, parmi ceux présents dans la liste, le plus rapidement possible et avec le maximum de précision. Au début de chaque exercice, les consignes étaient lues à l'utilisateur. Puis, pour chaque tâche, le mot à

trouver était énoncé. Il devait ensuite le trouver dans la liste. S'il sélectionnait un mot différent, sa réponse était enregistrée et considérée comme fausse. L'utilisateur ne pouvait pas la corriger, et la tâche suivante lui était proposée. A la fin de chaque session, le participant devait remplir un questionnaire SUS pour chaque disposition utilisée durant la session, et donner un ordre de préférence de ces dispositions.

1.2.4 Conception

Chaque participant devait réaliser une session comprenant deux exercices : un dans lequel les listes présentées aux participants comportaient quatre éléments, et un autre avec des listes composées de six éléments. Les deux exercices ont été menés consécutivement. Les douze participants étaient divisés en deux groupes de même taille. Chaque groupe a commencé par une exercice différent.

Durant chaque exercice, le participant devait réaliser cinq blocs. Chaque bloc correspondait à une disposition différente : Absolue en ligne, Absolue en colonne, Absolue en grille, Relative et par page. L'ordre de présentation des blocs a été contrebalancé par un carré latin. Chaque bloc était constitué de trois séries de tâches, réalisées l'une après l'autre. Chaque série comprenait autant de tâches qu'il y avait de positions possibles dans la liste (4 ou 6). Dans chaque série, la position du mot variait, de manière aléatoire, d'une tâche à l'autre de manière à tester toutes les positions possibles dans chaque série.

En résumé, lors de l'exercice avec 4 éléments par liste, chaque participant réalisait 5 blocs $\times$ 3 séries $\times$ 4 tâches, soit 60 tâches de sélection. De la même manière, chaque participant réalisait 90 tâches (5 blocs $\times$ 3 séries $\times$ 6 tâches) lors de l'exercice avec 6 éléments par liste. Au total, tous les participants ont réalisé 150 tâches de sélection. Nous avons ainsi recueilli pour cette expérimentation 1 800 essais.

1.2.5 Données collectées

Afin de pouvoir analyser les résultats quantitatifs, nous avons enregistré au cours de chaque session, l'ensemble des actions effectuées par l'utilisateur (pression, relâchement, et déplacements du doigt à l'écran), ainsi que les instructions données à

l'utilisateur (consignes, mot à sélectionner) et l'état de la liste qui lui était présentée (différents mots présents dans la liste et leur position). Enfin, nous avons aussi recueilli les réponses obtenues au questionnaire SUS, ainsi que l'ordre de préférence des dispositions données par chaque participant.

1.3 Résultats

Les figures 20, 21 et 23 présentent les principaux résultats de la première étude. Les barres verticales représentent les différentes dispositions testées : Absolue en ligne, Absolue en colonne, Absolue en grille, Relative et par page. Les barres rouges correspondent aux résultats obtenus pour les listes de 6 éléments, tandis que les barres bleues sont utilisées pour les listes de 4 éléments. Nous avons réalisé des tests de Wilcoxon ainsi que des analyses de Friedmann pour appuyer nos résultats dans le cas où ces derniers ne suivent pas une loi normale. Le seuil de confiance α a été fixé à 0,05.

L'ordre des exercices n'a eu aucune influence sur les résultats grâce au contrebalancement effectué.

1.3.1 Nombre d'éléments survolés

Nous définissons le nombre d'éléments survolés comme le nombre d'éléments que l'utilisateur parcourt avant qu'il ne valide sa sélection. Le mot validé est lui-même inclus dans le nombre d'éléments survolés. La figure 20 présente le nombre d'éléments survolés pour chaque disposition, en fonction du nombre d'éléments dans la liste.

La disposition « *par page* » est celle qui nécessite de survoler le moins d'éléments pour atteindre le mot souhaité : pour une liste de 4 éléments, l'utilisateur survole en moyenne 1,85 éléments lorsqu'il utilise la disposition « *par page* », alors qu'il est nécessaire de survoler environ 3 éléments avec les 4 autres dispositions (3 éléments, 2,9 éléments, 2,8 éléments et 3,1 éléments avec respectivement les dispositions Absolue en ligne, Absolue en grille, Absolue en colonne, et Relative). Cette différence sur le nombre d'éléments parcourus est significative entre la disposition par page et

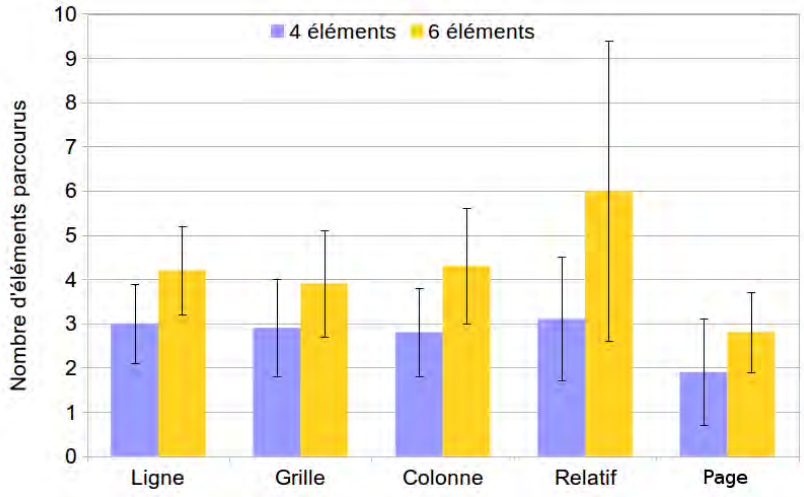

FIGURE 20 – Nombre d'éléments survolés

chaque disposition absolue ($W = 674$ pour la disposition en ligne, $W = 856$ pour la disposition en colonne et $W = 586$ pour la disposition en grille avec $p < \alpha$).

Ces différences sont encore plus marquées lorsque les participants utilisent une liste avec 6 éléments. Dans ce cas de figure, les participants ne survolent que 2,8 éléments avec la disposition par page alors qu'ils survolent environ 4 éléments avec la disposition absolue (4,2 éléments, 3,9 éléments et 4,3 éléments avec respectivement en ligne, en grille et en colonne) et 6 éléments avec la disposition relative. Les différences sont significatives à la fois entre la disposition par page et chaque disposition absolue ($W = 1783$ pour la disposition en ligne, $W = 1335$ pour la disposition en colonne et $W = 1910$ pour la disposition en grille avec $p < \alpha$), ainsi qu'entre la disposition par page et la disposition relative ($W = 1810$). Les différences sont également significatives entre les dispositions absolues et la disposition relative($W = 841$ pour la disposition en ligne, $W = 1229$ pour la disposition en colonne et $W = 874$ pour la disposition en grille avec $p < \alpha$). En revanche, il n'y a pas de différence significative entre chaque disposition absolue.

1.3.2 Temps de sélection

Le temps de sélection est le temps nécessaire pour sélectionner l'élément souhaité, calculé entre le moment où l'utilisateur pose son doigt sur l'écran pour commencer son parcours et le moment où il valide son choix en relâchant son doigt ou en effectuant un geste vers le bas (dans le cas de la disposition par page). La figure 21 présente le temps, exprimé en secondes, mis par l'utilisateur pour sélectionner le mot dans la liste pour chaque disposition et en fonction du nombre d'éléments dans la liste.

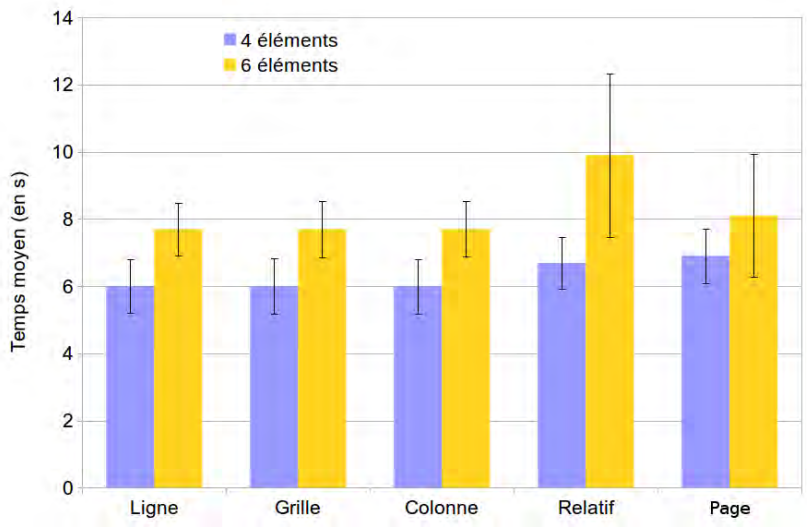

FIGURE 21 – Temps moyen de sélection en secondes

Pour les listes de 4 éléments, les trois catégories de disposition absolue présentent des valeurs quasiment similaires : 6,12 secondes, 6,05 secondes et 6,18 secondes pour respectivement la disposition en ligne, en grille et en colonne. En revanche, alors que l'utilisateur met en moyenne 6 secondes pour sélectionner un mot avec les dispositions absolues, il lui faut 6,69 secondes et 6,9 secondes pour sélectionner un mot avec les dispositions relatives et par page. Cette différence est significative entre chaque disposition absolue et la disposition relative ($W = 1663$ pour la disposition en ligne, $W = 2348$ pour la disposition en colonne et $W = 800$ pour la disposition en grille avec $p < \alpha$).

Pour des listes de 6 éléments, les dispositions absolues restent les plus efficaces pour atteindre l'élément souhaité (7,46 secondes, 7,38 secondes et 7,43 secondes pour respectivement les dispositions en ligne, en grille et en colonne). Dans le cas de l'utilisation de la disposition par page, les participants ont mis en moyenne 8,3 secondes pour sélectionner un élément. L'augmentation par rapport à la liste de 4 éléments est beaucoup plus importante pour la disposition relative : les participants ont eu besoin en moyenne de 9,8 secondes pour sélectionner son mot avec cette disposition. Cette différence de temps de sélection est significative entre les dispositions absolues et la disposition relative ($W = 1763$ pour la différence avec la disposition en ligne, $W = 1921$ avec la disposition en colonne et $W = 2247$ avec la disposition en grille avec $p < \alpha$).

Enfin, nous pouvons constater que les dispositions absolues sont moins dépendantes de la position du mot dans la liste (voir figure 22). Les dispositions relative et par page permettent d'accéder plus rapidement au premier élément de la liste, mais deviennent très rapidement moins efficaces pour les mots positionnés plus loin dans la liste.

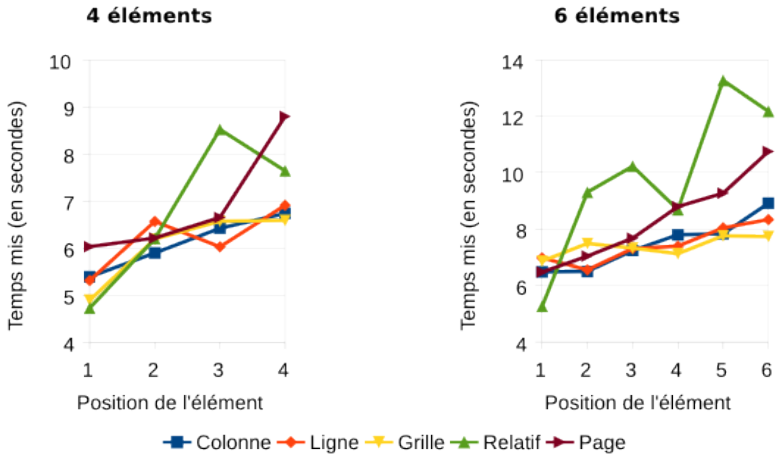

FIGURE 22 – Temps moyen de sélection en secondes en fonction de la position pour 4 éléments (à gauche) et pour 6 éléments (à droite)

1.3.3 Taux d'erreurs

Le taux d'erreurs est calculé comme étant le rapport des éléments sélectionnés de façon incorrecte divisé par le nombre total d'éléments sélectionnés. La figure 23 présente les taux d'erreurs, en pourcentage, pour chaque disposition et en fonction du nombre d'éléments dans la liste.

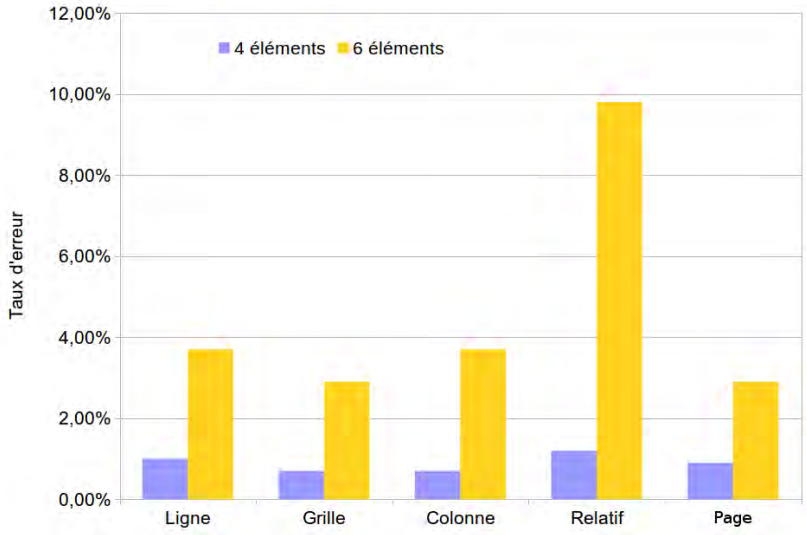

FIGURE 23 – Taux d'erreurs à la sélection

Pour les listes à 4 éléments, tous les taux d'erreurs sont faibles (moins de 2% pour chaque technique). En revanche, pour les listes de 6 éléments, La disposition absolue en grille et la disposition par page conservent des taux d'erreurs inférieurs à 3%, alors que les dispositions absolues en ligne et colonne ont des taux d'erreurs de 3,7% ; mais ces différences ne sont pas significatives. Seule la disposition relative a un taux d'erreurs très élevé avec 9,8%.

1.3.4 Retour utilisateur

La satisfaction des utilisateurs a été évaluée à travers les résultats recueillis avec le questionnaire SUS. Les participants ont donné leur avis sur les 3 dispositions proposées. Pour la disposition absolue, nous n'avons pas distingué les 3 catégories et

les participants n'ont rempli qu'un seul questionnaire SUS relatif à cette disposition. Comme le questionnaire était fait après chaque exercice, les scores sont donnés pour chaque technique en fonction du nombre d'éléments dans la liste (cf. figure 24).

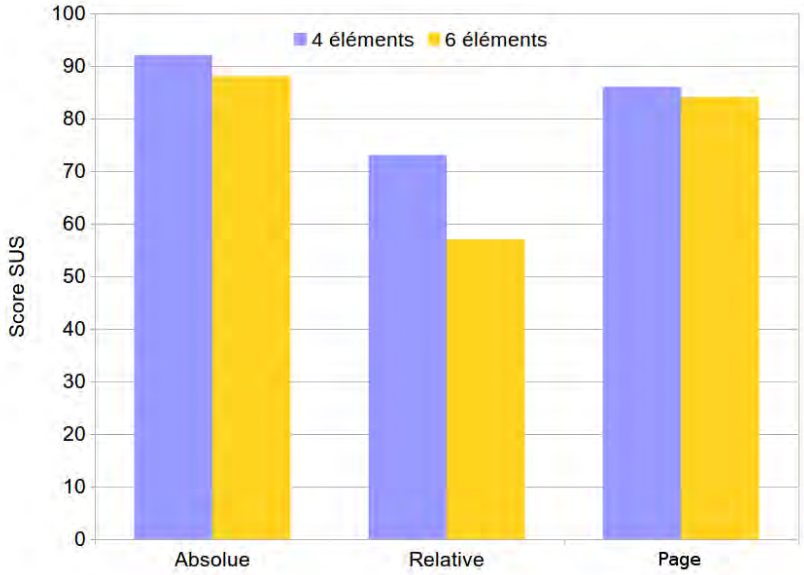

FIGURE 24 – Scores SUS

Nous pouvons constater que quelle que soit la disposition, les participants ont préféré utiliser des listes de 4 éléments. La disposition absolue ressort comme la disposition préférée des participants (avec des scores de 92 et 88 pour les listes de 4 et 6 éléments) alors que la disposition relative a été la moins appréciée (avec des scores de 73 et 57 pour les listes de 4 et 6 éléments).

1.4 Discussion

Le faible nombre d'éléments survolés avec la disposition par page peut s'expliquer par l'interaction utilisée qui est différente des deux autres dispositions. En effet, avec la disposition par page, l'utilisateur doit effectuer un geste pour passer d'un élément à un autre. Qu'il effectue un geste court ou long, il ne se déplace que d'un élément. Il est donc plus facile de parcourir les éléments un à un sans en oublier un. D'autre

part, de par sa configuration circulaire, l'utilisateur peut accéder au dernier élément directement à partir du premier élément, l'utilisateur peut alors accéder à la fin de la liste avec un nombre assez faible d'éléments parcourus. Par exemple, il est possible d'accéder au dernier élément de la liste en ne survolant que deux éléments : le premier et le dernier. Dans le cas des dispositions « *absolue* » et « *relative* », l'ensemble des éléments est présenté en même temps à l'écran, et il est possible que l'utilisateur passe trop vite sur un élément en effectuant un geste trop grand. Dans ce cas, si l'utilisateur saute l'élément souhaité, il est obligé de revenir en arrière, entraînant alors le survol d'un plus grand nombre d'éléments.

Cependant, le faible nombre d'éléments survolés avec la disposition par page ne se traduit pas par un gain de temps pour sélectionner le mot souhaité. Comme nous l'avons vu, les différentes dispositions absolues permettent à l'utilisateur d'accéder plus facilement au mot désiré. Ceci peut s'expliquer par l'interaction employée : l'utilisateur peut accéder à tous les éléments d'une disposition absolue en faisant glisser son doigt sur l'écran, alors qu'il est obligé d'effectuer un nouveau geste pour passer d'un élément à un autre avec la disposition par page. Par conséquent, bien que la disposition par page soit plus précise, l'interaction qui y est associée est trop coûteuse en temps pour que la vitesse d'exécution avec cette disposition soit bénéfique à l'utilisateur.

Enfin, le nombre d'éléments survolés plus élevé pour la disposition relative peut s'expliquer par la forme et la taille des zones de sélection des différents éléments de la liste. Les zones sont souvent moins larges que sur les autres dispositions ce qui rend le pointage d'une zone moins précis que pour les dispositions absolues et par page. Ce problème de précision empêche l'utilisateur de pointer rapidement l'élément souhaité. Du fait de la petite taille des zones, l'utilisateur risque de se déplacer sur l'élément suivant ou précédent, et ainsi nécessiter de la part de l'utilisateur de réajuster son geste à l'approche de l'élément souhaité. Cette imprécision va alors faire augmenter le nombre d'éléments survolés. Ce problème de précision de pointage se traduit aussi par une augmentation du temps de sélection, ainsi que d'une augmentation du taux d'erreurs de sélection. Enfin, le score SUS de cette disposition montre que les utilisateurs ont ressenti ce problème de précision. Pour eux, la disposition

relative avec une liste de 6 éléments n'est pas satisfaisante (score SUS de 57).

En résumé, la disposition « *absolue* » semble être la plus appropriée. En effet, elle est celle qui permet à l'utilisateur d'accéder le plus rapidement au mot souhaité tout en offrant un taux d'erreurs faible. La disposition « *par page* », qui était celle utilisée par le système de déduction de DUCK jusqu'à présent, permet de limiter le parcours des éléments, mais prend plus de temps du fait des nombreux gestes à réaliser pour atteindre un élément éloigné. Enfin, la disposition « *relative* » est inappropriée. En effet, elle est plus longue à utiliser que les deux autres, et engendre plus d'erreurs de sélection.

Par conséquent, pour la seconde étude, nous avons retenu la disposition absolue. D'autre part, nous ne gardons pour l'étude suivante que les dispositions absolue en ligne et en grille. Par rapport à la disposition en colonne, elles présentent l'avantage d'avoir des zones plus larges que la disposition en colonne, ce qui donne plus de précision à l'utilisateur au moment de sélectionner un élément dans la liste quelle que soit sa position. Enfin, nous avons vu que le temps consacré à la sélection et la navigation sur un élément est directement lié à la taille de la liste. De plus, les listes à 6 éléments engendrent un nombre d'erreurs bien plus important que les listes à 4 éléments. Comme il s'agit d'utiliser cette liste avec un système de déduction où le mot recherché est très fréquemment en début de liste, il ne semble pas opportun d'augmenter la taille de la liste. Par conséquent, pour les études suivantes, nous conserverons une liste de 4 éléments.

2 Retour audio

Au-delà de la façon dont les éléments sont disposés dans la liste, le retour audio offert à l'utilisateur est aussi très important. C'est en effet grâce à celui-ci que l'utilisateur va pouvoir prendre connaissance des différents mots présents dans la liste, et aussi les localiser au sein de la liste.

Pour augmenter la rapidité du retour d'information, des techniques sonores peuvent être utilisées à la place d'une synthèse de parole. Cela permet de raccourcir le temps

nécessaire à un retour audio. Zhao présente une approche de retour audio continu réalisé pour le Earpod[Zhao et al., 2007]. Le Earpod fournit aux utilisateurs des commentaires audio synchrones à leur saisie tactile. Il est conçu pour permettre aux utilisateurs de découvrir les menus et les listes à leur propre rythme. Chaque interaction faite par l'utilisateur est sonifiée, ce qui rend un retour audio plus précis qu'un retour visuel. Ce système, selon eux, devient moins efficace lorsqu'une imbrication apparaît ou que les menus et listes sont de grandes tailles. Enfin, lorsque les éléments sont très différents mais peu nombreux, l'un des choix les plus courant est l'utilisation de « *earcons* » [Helle et al., 2001] : un earcon, de façon similaire à une icône, utilise un son simple pour permettre de distinguer une information rapidement (par exemple un bip d'alarme pour une erreur, un bruit de papier déchiré pour notifier une suppression, ...). Une étude faite par Walker[Walker and Kogan, 2009] compare les earcons à une approche basée sur la synthèse vocale : compte tenu des résultats, la synthèse vocale est plus efficace quand elle est couplée avec earcons et spearcons. Les spearcons, comme leurs homologues les earcons, sont des sons faits pour distinguer les différents éléments. Contrairement aux earcons, cependant, ils sont constitués d'une synthèse vocale qui est accélérée au point d'être trop rapide pour être pleinement reconnaissable.

Cependant, ces techniques ne sont pas applicables à notre cas d'étude. En effet, les techniques à base de sons ne sont utilisables que dans le cas où les éléments sont connus de l'utilisateur. Dans notre cas, la liste de mots change constamment en fonction de la saisie de l'utilisateur. Par conséquent, il est impossible d'associer un son différent à chaque mot possible de la liste. C'est pourquoi, dans le cas de notre liste de déduction, nous sommes contraints de conserver une synthèse vocale qui reste compréhensible par l'utilisateur.

Lors de l'utilisation d'une synthèse vocale avec une liste de mots, le retour qui est généralement utilisé consiste à prononcer le mot au moment où l'utilisateur survole le mot avec son doigt. Cela permet ainsi à l'utilisateur de savoir sur quel élément il se trouve. En revanche, cette technique ne permet pas à l'utilisateur d'avoir une connaissance globale des éléments présents dans la liste. Sans cette connaissance, l'utilisateur est obligé d'explorer séquentiellement la liste pour en connaître les dif-

férents éléments. Si l'élément recherché est en début de liste, cette exploration n'est pas trop coûteuse en temps. En revanche, si l'élément recherché est en fin de liste ou n'est pas présent dans la liste, l'utilisateur va perdre beaucoup de temps à rechercher le mot souhaité.

Ce travail a pour but d'étudier l'intérêt de présenter l'ensemble des éléments de la liste à un utilisateur avant que celui-ci ne l'utilise. Notre hypothèse est que si l'utilisateur a connaissance des mots présents dans la liste ainsi que leur position, il pourra alors plus rapidement sélectionner le mot désiré. Ceci ne sera possible que s'il connaît la disposition des mots à l'écran et n'aura pas besoin de commencer par le premier élément de la liste.

2.1 Retours audio étudiés

Dans cette expérience, nous évaluons et comparons deux types de stratégies de restitution de l'information :

— D'une part, le retour « *descriptif* » : c'est la stratégie habituellement utilisée. Elle consiste à prononcer seulement le mot lorsque l'utilisateur le survole avec son doigt ;

— D'autre part, le retour « *prescriptif* » : le retour audio est produit avant même que l'utilisateur commence à explorer la liste. Dès que la liste apparaît, l'ensemble des mots la constituant est lu à l'utilisateur. En plus de ce retour vocal initial, l'utilisateur a, en plus, le retour descriptif tout au long de sa phase d'exploration de la liste.

Notre principale hypothèse est que si l'utilisateur connaît préalablement l'ensemble des mots constituant la liste, il ne sera pas obligé de parcourir l'ensemble de la liste pour trouver le mot recherché et pourra plus rapidement y accéder en débutant son exploration à l'endroit où il estime trouver le mot recherché. Si la phase d'exploration est moins longue, l'utilisateur devrait alors sélectionner son mot plus rapidement. Notre première hypothèse (H1) est donc que l'utilisateur aura a parcourir moins d'éléments de la liste pour trouver son mot si la liste lui est annoncée initialement. Notre seconde hypothèse (H2) est que si l'utilisateur parcourt moins

d'éléments, il sélectionnera son mot plus rapidement.

Enfin, lors de cette étude, nous avons choisi d'étudier les deux stratégies d'oralisation avec deux dispositions de mots : absolue en ligne et absolue en grille. Avec seulement 4 mots, nous faisons l'hypothèse que la disposition sous forme de grille peut permettre à l'utilisateur d'accéder plus facilement aux 4 éléments de la liste avec une zone présente à chaque coin de l'écran. En pointant à proximité des coins de l'écran du smartphone, l'utilisateur devrait faire moins d'erreurs de sélection que dans une liste en ligne où les éléments en deuxième et troisième position sont entourés par d'autres éléments (les éléments 1 et 4 étant eux sur les extrémités de l'écran).

2.2 Protocole

2.2.1 Participants

12 participants (3 femmes et 9 hommes), d'une moyenne d'âge de 25 ans, avec vision normale ou corrigée ont participé à notre étude. Les participants avaient les yeux bandés pour s'assurer qu'ils ne pouvaient rien voir au cours de l'expérimentation.

2.2.2 Dispositif

Les participants ont utilisé le même smartphone que dans la première étude. Le système de synthèse vocale était aussi identique. La liste était constituée uniquement de 4 éléments affichés avec la disposition absolue. Les mots choisis étaient des noms de fruits facilement différentiables les uns des autres. Rien n'était affiché à l'écran durant l'expérience, rendant ainsi chaque agencement d'éléments invisible à l'utilisateur.

2.2.3 Procédure

Les participants avaient pour tâche de sélectionner le mot demandé dans la liste proposée, aussi vite que possible, et avec le maximum de précision. Pour chaque tâche, le mot à sélectionner était énoncé à l'utilisateur. Celui-ci devait alors le trouver le plus rapidement possible dans la liste. A la différence de la première expérimentation, le mot demandé pouvait être absent de la liste présentée : l'utilisateur devait dans

ce cas l'identifier comme absent. Dans tous les cas, si le participant sélectionnait un mot différent, la réponse était enregistrée et considérée comme fausse. L'utilisateur n'avait pas la possibilité de la corriger, et l'exercice passait à la tâche suivante. A la fin de chaque session, le participant devait remplir un questionnaire SUS pour chaque stratégie utilisée durant la session.

2.2.4 Conception

Chaque participant a réalisé quatre exercices consécutivement, un avec chaque couplage possible entre stratégie (descriptive ou prescriptive) et disposition (en ligne ou en grille). Les douze participants ont été répartis en quatre groupes de même taille de manière à contrebalancer l'ordre de réalisation des exercices. L'ordre des exercices pour chaque groupe a été choisi au moyen d'un carré latin pour éviter les effets d'apprentissage ou de fatigue liés notamment à la disposition utilisée.

Chaque exercice était constitué de trois blocs de tâches, exécutés consécutivement. Chaque bloc comprenait cinq tâches, une pour chacune des quatre positions possibles dans la liste, plus une tâche où le mot demandé était absent de la liste. De façon similaire à notre première expérience, dans chaque bloc, l'ordre des positions a été tiré aléatoirement.

Au total, nous avons enregistré 720 essais.

2.2.5 Données collectées

Les mêmes types de données que lors de la première expérimentation ont été enregistrées au cours de cette deuxième expérience.

2.3 Résultats

2.3.1 Éléments parcourus

Le principal résultat de notre deuxième étude concerne le nombre d'éléments parcourus pour atteindre le mot désiré (voir Figure 25). Lorsque l'utilisateur reçoit un retour de type « *prescriptif* », il parcourt 2 éléments en moyenne, contre 3,73

éléments en moyenne avec le retour « *descriptif* » ($W = 1362$, et $p < \alpha$). Ce résultat confirme donc notre première hypothèse H1. L'avantage du retour de type prescriptif est encore plus important lorsque le mot est absent de la liste. Dans ce cas de figure, les utilisateurs ne parcourent que 3,28 éléments avec le retour prescriptif alors qu'ils explorent en moyenne 6,18 éléments avec le retour descriptif.

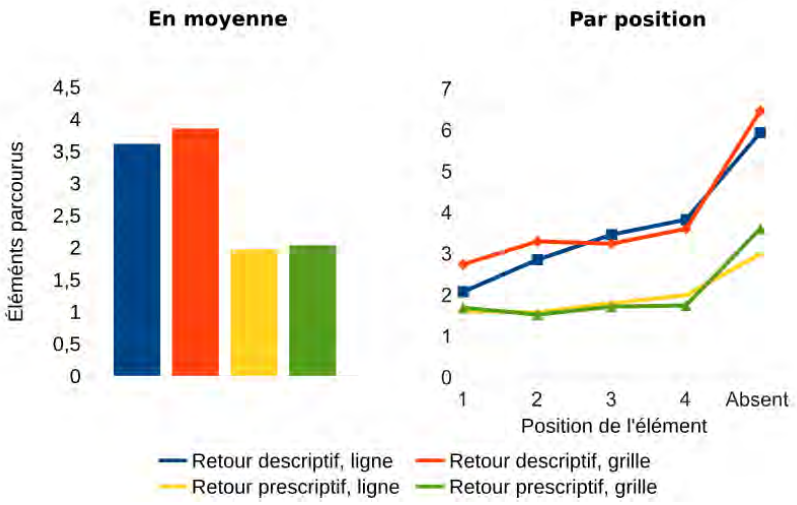

FIGURE 25 – Éléments parcourus

D'autre part, dans le cas d'un retour prescriptif, il n'y a pas de différence entre les deux types de disposition utilisée : Les participants ont parcouru 1.97 éléments avec la disposition en ligne et 2.03 éléments avec la disposition en grille. Il n'y a pas non plus de différence significative entre les deux dispositions lors de l'utilisation du retour descriptif.

2.3.2 Temps

La figure 26 présente les résultats de temps de sélection pour chaque type de retour, couplé à chaque disposition.

En moyenne, les participants ont pris plus de temps pour sélectionner un élément avec le retour prescriptif (5,59 secondes) qu'avec un retour descriptif (5,25 secondes), mais cette différence est non significative. Pour expliquer cette différence de temps par

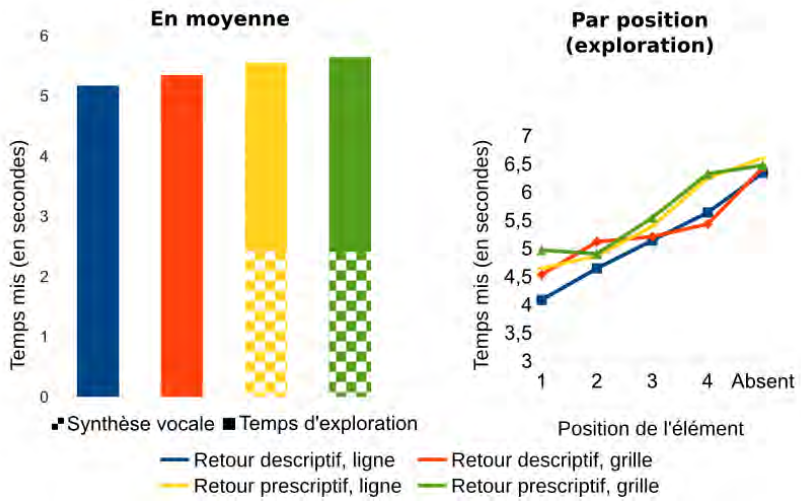

FIGURE 26 – Temps de sélection

rapport au nombre d'éléments survolés, nous avons décomposé le temps de sélection en deux phases pour le retour prescriptif : dans un premier temps, le temps pris par la synthèse vocale pour énoncer la liste à l'utilisateur, puis ensuite le temps nécessaire à l'utilisateur pour aller sélectionner le mot souhaité. On peut constater que le temps de sélection avec ce type de retour est inférieur au temps de sélection avec le retour descriptif. Cependant le temps nécessaire à la synthèse vocale au lancement de la liste est trop conséquent (2,4 secondes), et n'est pas compensé par le gain de temps lors de la sélection pour que ce retour audio soit meilleur que le retour descriptif.

D'autre part, pour les deux types de retour, la disposition en ligne permet à l'utilisateur d'accéder plus rapidement à l'élément souhaité que la disposition en grille : dans le cas du retour descriptif, les participants ont mis en moyenne 5.17 secondes pour la disposition en ligne contre 5.34 secondes pour la disposition en grille ; dans le cas du retour prescriptif, le temps de sélection est de 5.54 secondes et 5.64 secondes pour respectivement les dispositions en ligne et grille.

La figure 26 de droite présente les temps de sélection observés en fonction de la position dans la liste du mot à sélectionner. On peut constater que quelle que soit la position dans la liste, le retour descriptif permet toujours aux utilisateurs de

sélectionner plus rapidement le mot souhaité. On peut également observer que quelle que soit la position dans la liste, la disposition en ligne permet à l'utilisateur de toujours sélectionner plus rapidement le mot dans la liste.

2.3.3 Erreur

La figure 27 présente les taux d'erreurs d'une part en fonction du retour, et d'autre part en fonction de la disposition utilisée. Dans le cas du retour descriptif, le taux d'erreurs est de 1,65% pour la disposition en ligne, contre 2,32% pour la disposition en grille. Dans le cas du retour prescriptif, le taux d'erreurs observé est de 4,48% pour la disposition en ligne alors qu'il n'est que de 0,52% pour la disposition en grille.

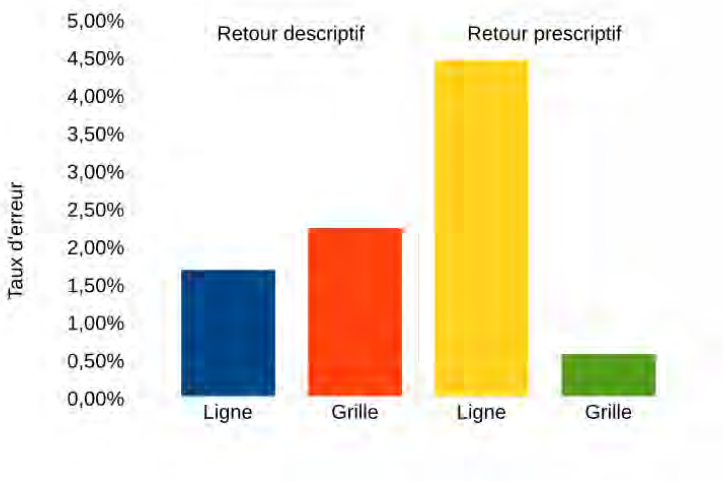

FIGURE 27 – Erreur par technique

2.3.4 Questionnaire SUS

La condition préférée des participants a été le couplage entre retour descriptif et disposition en ligne, avec un score SUS de 82. En revanche, le retour descriptif couplé à la disposition en grille a obtenu le plus faible score SUS avec 73. Dans le cas du retour prescriptif, le même score a été obtenu quelle que soit la disposition avec 79.

2.4 Discussion

Même si le retour prescriptif permet à l'utilisateur d'explorer moins d'éléments au moment de la sélection, le temps nécessaire pour énoncer l'ensemble de la liste au départ est pénalisant : ce temps passé à écouter les différents mots de la liste n'est pas compensé par le temps gagné lors de l'exploration de la liste. Par conséquent, l'utilisateur est plus rapide pour sélectionner un mot avec un retour descriptif.

D'autre part, nous avons vu que la disposition des éléments en ligne était plus efficace qu'une disposition en grille. Les participants ont expliqué cette différence par une plus grande difficulté à identifier l'ordre des mots de la liste dans le cas d'une disposition en grille. Ils ont plus de facilité à explorer une liste dont les éléments sont disposés en ligne. Ces commentaires sont confirmés par le score SUS plus élevé obtenu par le couplage retour descriptif avec la disposition en ligne.

Pour la suite de notre étude, nous avons donc conservé la disposition en ligne avec un retour audio descriptif.

3 Technique de validation

La dernière étude que nous avons conduite sur l'interaction avec les listes concerne la validation d'un mot, une fois que celui-ci est sélectionné par l'utilisateur. Le principal problème rencontré lors de cette phase de validation est la précision au moment de la validation : comme nous l'avons vu précédemment, l'utilisateur va sélectionner son mot dans la liste en explorant la liste au moyen d'un doigt sur l'écran. Lorsque le doigt survole un élément, ce dernier est énoncé par la synthèse vocale. Une fois que l'utilisateur a son doigt positionné sur le mot qu'il souhaite, il doit effectuer la validation de cette sélection tout en conservant son doigt appuyé sur l'élément en question. Si lors de la validation son doigt se déplace sur un autre élément, le mot sélectionné ne sera alors pas le bon. Il est donc nécessaire d'avoir une action de validation qui puisse se faire sans affecter la sélection en cours.

Pour cela, nous avons étudié quatre interactions de validation différentes :

— Une validation par « *relâchement* » : l'utilisateur relâche son doigt de l'écran

pour confirmer son choix. Le dernier élément survolé par le doigt est alors celui qui est validé.

— Une validation par « *frappe à deux doigts* » : l'utilisateur utilise un second doigt pour frapper sur l'écran [Kane et al., 2008]. Le mot sélectionné est celui se trouvant sous le premier doigt au moment où l'utilisateur tape avec son second doigt.
— Une validation par « *double frappe* » : l'utilisateur valide son choix en effectuant une double frappe avec un seul doigt (semblable à un double-clic).
— Une validation par « *ligne brisée* » : l'utilisateur fait glisser son doigt sur l'écran pour sélectionner un mot. Pour valider sa sélection, l'utilisateur doit effectuer un angle droit par rapport à son tracé.

3.1 Protocole

3.1.1 Participants

12 participants (5 femmes et 7 hommes, âge moyen = 28,5) avec vision normale ou corrigée ont participé à notre étude. Les participants avaient les yeux bandés pour s'assurer qu'ils ne pouvaient pas voir l'écran.

3.1.2 Dispositif et procédure

Les participants ont utilisé le même matériel que pour l'expérience précédente. Le corpus de mots utilisé était aussi identique à l'expérience précédente.

Selon nos observations des deux études précédentes, nous avons choisi de conserver une disposition en ligne contenant quatre éléments, avec un retour de type « *descriptif* ».

La tâche demandée aux participants était aussi identique à celle des deux expériences précédentes. Les participants avaient un questionnaire SUS à remplir à chaque fin de session.

3.1.3 Conception

L'ensemble de l'expérience a été réalisé en une seule session pour tous les participants, constituée de quatre exercices, exécutés consécutivement. Chaque exercice était consacré à une interaction de validation, contrebalancé par un carré latin. Chaque exercice était constitué de deux séries de tâches, exécutées l'une après l'autre. Chaque série comprenait quatre tâches de sélection. Pour chaque tâche, la position du mot à sélectionner était choisie aléatoirement. Cependant, dans chaque série, chacune des positions possibles dans la liste était testée par le biais d'une tâche.

Nous avons collecté 384 essais au total.

3.1.4 Données collectées

Les mêmes types de données que lors des deux premières expérimentations ont été enregistrées au cours de cette expérience.

3.2 Résultats

3.2.1 Temps

Nous avons d'abord calculé le temps nécessaire pour valider un élément, c'est-à-dire le temps passé entre le moment de la fin de lecture du mot à sélectionner et le moment où l'utilisateur valide sa sélection.

La figure 28 présente les résultats pour chaque technique. Les participants ont obtenu des temps de validation assez proches pour les techniques par relâchement et frappe à deux doigts avec respectivement 5,65 et 5,66 secondes. La technique de double frappe nécessite 6,16 secondes, alors que la technique ligne brisée apparaît comme la plus lente avec en moyenne 6,7 secondes pour valider le mot recherché. Cependant les différences de temps entre ces différentes techniques ne sont pas significatives.

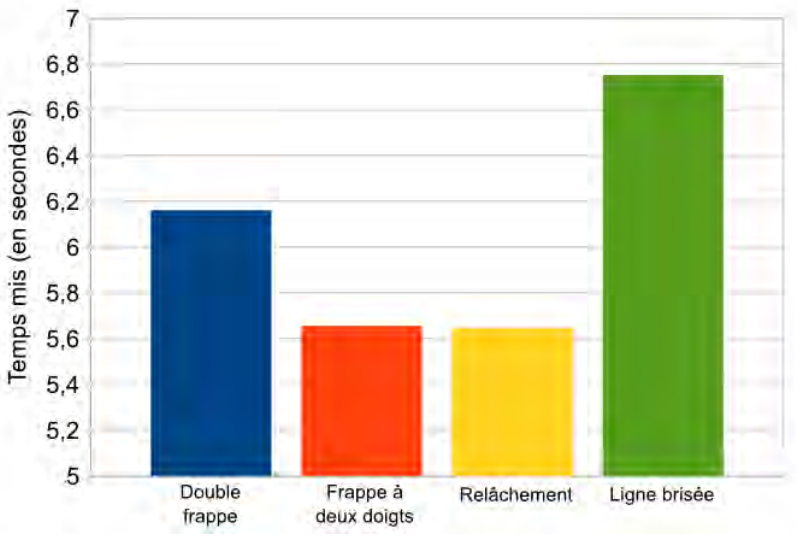

FIGURE 28 – Temps par geste

3.2.2 Erreur

Nous avons également calculé le taux d'erreurs pour chaque technique de validation (figure 29).

Les participants ont réalisé en moyenne 3,1% d'erreur avec les techniques de relâchement et double frappe, contre 6,2% et 9,3% avec les techniques de frappe à deux doigts et ligne brisée.

3.2.3 Résultats qualitatifs

Les scores au questionnaire SUS donnent les techniques de frappe à deux doigts et relâchement comme les plus appréciées par les participants (respectivement 81 et 85). En revanche, les techniques ligne brisée et double frappe obtiennent un score SUS inférieur à 70. D'autre part, ces scores sont confirmés par l'ordre de préférence donné par les participants dont cinq d'entre eux ont classé relâchement en première position et quatre autres ont classé la frappe à deux doigts en première position.

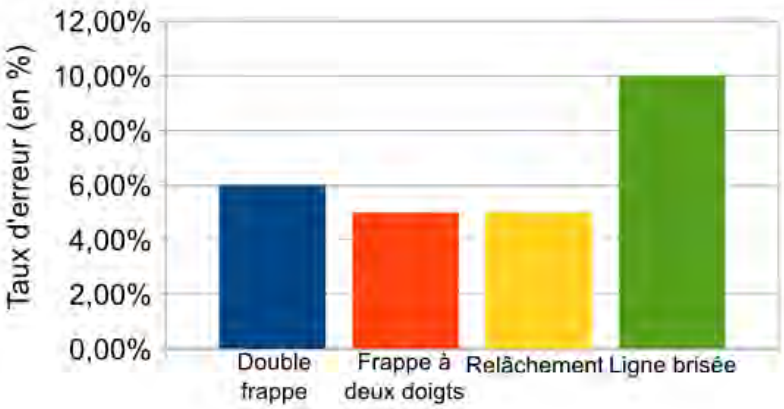

FIGURE 29 – Erreur par geste

3.3 Discussion

La technique ligne brisée apparaît comme inadaptée pour effectuer une validation dans une liste de mots par des utilisateurs non-voyants. En plus de nécessiter plus de temps pour valider le mot, celle-ci engendre un taux d'erreurs élevé. Ces erreurs peuvent s'expliquer par la difficulté à réaliser un geste perpendiculaire à l'exploration dans la liste sans un retour visuel. Pour que la validation soit faite de manière précise, il faut pouvoir clairement distinguer la partie du geste de l'utilisateur liée à la phase d'exploration et celle liée à la validation. Ainsi, si le début du geste de validation est mal détecté, la sélection de l'utilisateur peut alors être modifiée et ainsi engendrer une erreur. De la même manière, la technique frappe à deux doigts engendre aussi un nombre d'erreurs élevé. Notre hypothèse est que l'utilisateur, en appuyant un second doigt à l'écran, modifie légèrement la position du premier doigt. Cette modification peut alors entraîner un changement dans la sélection effectuée par l'utilisateur.

Les deux techniques préférées par les utilisateurs sont aussi celles qui leur ont permis de valider avec le plus de précision leur sélection. Dans les deux cas, la sélection de l'utilisateur est figée au moment où celui-ci relève son doigt de l'écran. Dans le cas de la technique relâchement, cela suffit à valider la sélection. Dans le cas de la double frappe, l'utilisateur doit retaper sur l'écran pour confirmer cette vali-

dation. Cette dernière action demande donc à l'utilisateur un temps supplémentaire pour la validation de son mot (200ms), mais apporte en contre partie une précision supplémentaire de la validation à l'utilisateur. En effet, dans le cas de la technique relâchement, si l'utilisateur relève par mégarde son doigt de l'écran, le dernier mot survolé est alors validé. Dans le cas de double frappe, avec la confirmation par cette interaction supplémentaire, l'utilisateur peut se permettre de relever le doigt de l'écran et le reposer pour reprendre sa sélection.

4 Synthèse

Il ressort de ces études que la meilleure façon de présenter les mots de la liste à l'utilisateur est de tous les afficher en même temps sur l'écran. L'utilisateur peut y accéder plus rapidement que s'ils sont sur des pages différentes, et avec plus de précision que lors d'un affichage relatif à la position du doigt de l'utilisateur.

D'autre part, dans le cas de liste courte, le meilleur retour audio consiste à lire les mots seulement lorsqu'ils sont survolés. Même si lire l'ensemble des mots au démarrage de la liste permet à l'utilisateur d'explorer sa liste et trouver son mot plus rapidement, le temps nécessaire pour lire tous les mots au début de la liste est trop coûteux.

Enfin l'étude sur la validation montre que la meilleure manière de valider un mot avec précision consiste à relever le doigt de l'écran une fois que celui-ci est positionné sur l'élément souhaité. Il peut être accompagné d'une phase de confirmation, avec une double frappe, pour sécuriser les actions de l'utilisateur et ainsi distinguer un relâchement involontaire d'un relâchement synonyme de validation.

Par conséquent, à partir des trois études que nous venons de présenter, nous avons choisi de conserver pour la liste de mots de notre système DUCK, une disposition des éléments en ligne avec un retour descriptif et une validation simplement en relevant le doigt de l'écran. C'est cette combinaison qui s'avère être la plus rapide pour sélectionner un élément dans la liste.

Les temps de validation obtenus lors de ces expériences sont plus élevés que les

temps de la phase de validation obtenus lors de la première expérimentation avec le système DUCK. Cependant, nous ne pouvons comparer directement ces résultats car les tâches demandées aux participants ne sont pas du même type entre les deux évaluations. En revanche, si nous nous référons aux résultats obtenus par la disposition par page de notre première étude, qui était la disposition utilisée jusqu'à présent par la liste du système DUCK, nous pouvons constater que notre solution retenue au bout des trois études est plus rapide : en effet, il fallait en moyenne 6,9 secondes pour valider un mot avec la disposition par page, alors que les participants n'ont besoin que de 5,65 secondes avec la disposition et les interactions finalement retenues. Ainsi nous pensons pouvoir réduire le temps de validation lors de la saisie avec DUCK grâce à cette nouvelle disposition et ces nouvelles interactions.

Chapitre V

La gestion des mots courts

Comme nous avons pu le constater au chapitre III, la phase de validation nécessite trop de temps pour que l'utilisateur tire profit du système DUCK pour les mots de 4 caractères ou moins. Même si l'étude précédente nous permet de réduire ce temps de validation grâce à une interaction plus adaptée, la phase de validation restera une étape nécessitant un temps difficilement compensable pour les mots courts. Or, dans la langue française, environ 47% des mots usuels sont constitués d'au plus quatre lettres [1]. Cette importante fréquence d'utilisation des mots courts dans la langue française nous a donc amené à étudier une manière de saisir ces mots afin de réduire le temps de saisie.

Zhai et Kristensson[Zhai and Kristensson, 2003] ont étudié la saisie des mots les plus fréquemment saisis en utilisant des gestes sténographiques pour entrer les mots les plus courants parmi les mots de la langue anglaise. Le geste proposé pour chaque mot correspond au chemin à réaliser sur le clavier pour passer sur chaque caractère du mot. De ce fait, pour apprendre les différents gestes, l'utilisateur peut s'aider de l'affichage de la disposition des caractères à l'écran pour réaliser le geste correspondant au mot. Peu à peu avec l'apprentissage, l'utilisateur n'a plus besoin de regarder la disposition des caractères et est capable d'effectuer le geste rapidement pour saisir son mot.

1. selon nos mesures sur le projet Gutenberg, `https://www.gutenberg.eu.org/`, consulté en septembre 2016

Dans notre cas, cette solution n'est pas envisageable car un utilisateur novice ne peut pas se servir de l'affichage de la disposition des caractères pour apprendre les différents gestes. Pour accéder rapidement aux mots courts les plus fréquents, nous avons choisi de nous baser sur l'utilisation de liste. Une liste différente est associée à chaque caractère. Ainsi, pour chaque caractère nous proposons les 8 mots courts les plus fréquents commençant par cette lettre. Cette liste est statique : à chaque fois que l'utilisateur saisit un caractère donné, c'est la même liste de mots débutant par ce caractère qui lui est proposée. Ceci permet à l'utilisateur de pouvoir mémoriser petit à petit les différentes listes et accéder plus rapidement au mot désiré.

Notre étude se base sur la manière d'accéder à ces différentes listes pour que l'utilisateur puisse rapidement saisir ces mots courts. Afin de rendre notre technique facilement intégrable à DUCK, nous avons essayé de conserver une interaction proche de celle déjà utilisée pour la saisie avec le système DUCK. Ainsi, l'utilisateur entre le premier caractère du mot de la même manière qu'il saisit le premier caractère pour n'importe quel mot avec le système DUCK. Puis, à la place de frapper approximativement les caractères suivants, il valide ce caractère pour accéder à la liste de mots.

Afin de faciliter l'utilisation de cette liste, les mots sont disposés à l'écran de la même manière que les mots proposés dans la liste de déduction. L'ensemble de l'écran est utilisé par la liste et chaque mot présent dans la liste à une surface d'affichage identique. La liste utilise une disposition absolue en ligne : les éléments sont alignés de façon contiguë le long de la largeur de l'écran, tout en prenant la hauteur maximale de l'écran. Lorsqu'un utilisateur parcourt la liste, il doit garder son doigt appuyé sur l'écran pendant que les éléments situés au-dessous lui sont lus. Au relâchement du doigt, le mot situé en dessous est validé. Enfin, afin de faciliter la mémorisation, les mots sont toujours présentés dans le même ordre.

Afin de déterminer le corpus des mots courts utilisées dans les différentes listes, nous avons utilisé une heuristique simple basée sur la répartition fréquentielle des mots courts en français. Nous avons sélectionné les plus usuels, en en prenant au maximum 8, débutant par le même premier caractère. D'autre part, seuls 22 des 26 caractères de l'alphabet correspondent à la première lettre d'un mot court les plus

utilisés. Donc, il n'y a que 22 listes de mots courts.

Ce chapitre est axé sur l'étude de l'interaction la plus adaptée pour passer d'un mode de saisie classique avec DUCK à la saisie spécifique d'un mot court. Le but est de trouver une interaction qui soit à la fois rapide pour accéder à la liste de mots courts, et d'autre part, qui ne présente pas d'ambiguïté possible avec le reste de la saisie avec le système DUCK. C'est pourquoi, la validation du premier caractère ne pouvait pas se faire simplement avec le relâchement du doigt car cette interaction valide déjà le premier caractère saisi, quelle que soit la longueur du mot à saisir. De ce fait, nous avons choisi de comparer deux autres techniques d'interaction permettant de basculer d'un mode de saisie classique à la sélection d'un mot court.

Avec la première technique que nous proposons, appelée validation « *à double frappe* » (Figure 30), l'utilisateur parcourt le clavier en glissant son doigt pour localiser la lettre qu'il recherche. Il relâche ensuite son doigt pour valider son choix, puis appuie avec deux doigts sur l'écran pour afficher la liste des mots courts.

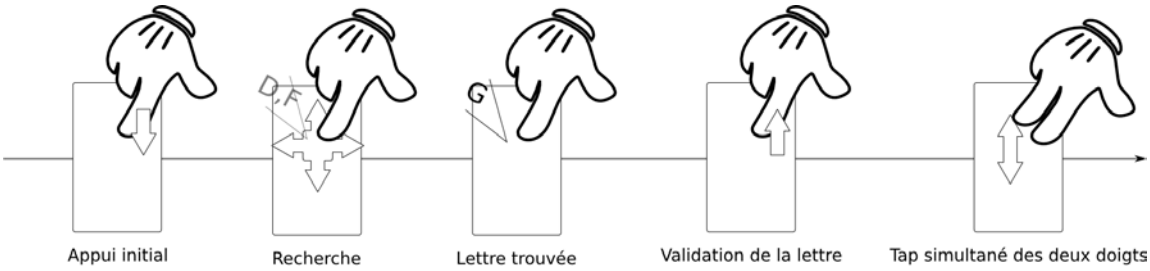

FIGURE 30 – Le mode de validation à frappe double

Dans la seconde technique que nous souhaitons étudier, appelée validation « *à deux doigts* » (Figure 31), l'utilisateur parcourt le clavier en glissant son doigt de la même façon, mais à la place de relever directement son doigt de l'écran, il appuie un deuxième doigt à l'écran. La liste de mots courts est alors affichée au moment où il relève ses deux doigts de l'écran.

Afin d'évaluer la pertinence de l'utilisation d'une liste de mots courts, ainsi que l'interaction la plus appropriée pour y accéder, nous avons choisi de comparer nos

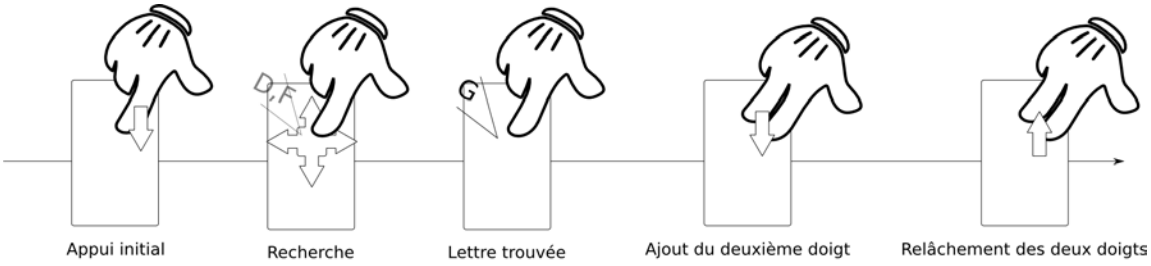

FIGURE 31 – Le mode de validation à deux doigts

deux techniques d'interaction à une troisième, qui conserve le principe d'exploration douloureuse : l'utilisateur doit saisir chaque caractère du mot en conservant son doigt appuyé à l'écran. Pour valider un caractère, il frappe avec un deuxième doigt sur l'écran pour chaque nouveau caractère à entrer (voir Figure 32). Une telle approche offre une saisie libre, non limitée aux mots courts, tout en étant non-intrusive par rapport au fonctionnement traditionnel du système DUCK. Nous avons appelé cette technique « *glisser-taper* ». Cette technique nous sert de technique contrôle dans notre expérience car elle se rapproche de la saisie proposée avec le clavier VODKA, tout en permettant son fonctionnement sur le clavier DUCK.

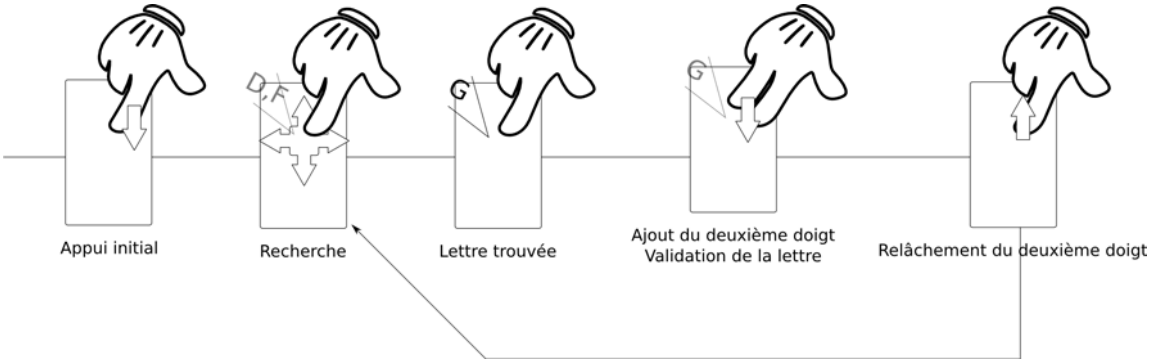

FIGURE 32 – La saisie glisser-taper

1 Protocole

1.1 Hypothèses

Nous avons choisi d'utiliser la saisie glisser-taper comme base pour notre étude comparative suite au constat fait dans l'étude préliminaire (voir chapitre III), où VODKA était plus efficace que DUCK pour les mots courts (mots de moins de 4 caractères).

Nous basons notre principale hypothèse sur l'idée que la réduction du nombre d'actions pour saisir un mot court permettra à nos techniques d'interaction de réduire le temps de saisie d'un mot court.

1.2 Participants

Nous avons sélectionné 12 participants (4 femmes et 8 hommes, âge moyen = 24.7 ans) avec une vision normale (ou corrigée) pour notre étude. Tous les participants ont eu les yeux bandés pour s'assurer qu'il leur était impossible de voir l'écran. Tous avaient une bonne maîtrise (par une pratique quotidienne) des périphériques mobiles et des smartphones. Parmi eux, six ont affirmé avoir une bonne maîtrise du clavier AZERTY de part une expérience de jeu très importante. Aucun de ces participants n'était déficient visuel.

1.3 Dispositif

Les participants ont utilisé un smartphone Samsung Galaxy SII. Le dispositif a une résolution de 306 ppm pour un écran de 136,6 mm × 70,6 mm, un processeur ARM Cortex-A9 MPCore Quad core cadencé à 1.4 GHz et un système d'exploitation Android 4.3. Ils ont utilisé leurs doigts pour naviguer à travers les éléments. Ces derniers étaient prononcés à l'aide du service Google Translate. La longueur du fichier audio était ensuite minimisée pour permettre un retour audio aussi rapide que possible, réduisant ainsi tout retard. Rien n'était affiché à l'écran durant l'expérience, rendant ainsi les éléments invisibles à l'utilisateur.

1.4 Procédure

Les participants avaient pour tâche de saisir un mot court, d'au plus quatre lettres, aussi rapidement que possible. En début d'exercice, les consignes étaient lues au sujet, puis pour chaque tâche, le mot à saisir était prononcé. L'utilisateur devait alors le saisir le plus rapidement possible avec la technique demandée. S'il saisissait un mot différent, sa réponse était considérée comme fausse. Une fois la session terminée, le participant devait remplir un questionnaire SUS pour chaque technique, et préciser celle qu'il jugeait la plus adaptée parmi les interactions qu'il avait testées, en les classant par ordre de préférence.

1.5 Conception

Chaque participant a effectué une session composée de trois exercices, un pour chaque interaction testée : validation « *à frappe double* », validation « *à deux doigts* », et « *glisser-taper* ». Les trois exercices ont été réalisés les uns après les autres. Pour cela, nous avons réparti aléatoirement les douze participants en trois groupes de même taille. Chaque groupe réalisait les trois exercices dans un ordre différent, défini par un carré latin.

Pour chaque exercice, le participant avait 44 tâches de saisie d'un mot à effectuer. Ainsi, dans chaque exercice, l'utilisateur avait à saisir deux mots pour chacune des listes de mots courts. Les mots à recopier étaient choisis aléatoirement parmi ceux présents dans la liste des mots courts.

Nous avons ainsi collecté 1 584 essais.

1.6 Données collectées

Nous avons enregistré au cours de chaque session, l'ensemble des actions effectuées par l'utilisateur (pression, relâchement, et déplacements du doigt à l'écran), ainsi que les instructions données à l'utilisateur (consignes, mot à saisir) et l'état du clavier et de la liste qui lui était présentés (les différents mots présents dans la liste et leur position). Enfin, nous avons aussi recueilli les réponses obtenues au questionnaire

SUS, ainsi que l'ordre de préférence des techniques données par chaque participant.

2 Résultats

Nous avons réalisé des tests de Wilcoxon ainsi que des analyses de Friedmann pour appuyer nos résultats dans le cas où ces derniers ne suivent pas une loi normale. Le seuil de confiance α a été fixé à 0,05.

2.1 Temps

Nous mesurons le temps mis par l'utilisateur entre le moment où il entre la première lettre du mot, et le moment où le mot est validé.

La figure 33 montre les résultats obtenus avec chaque technique lors de la saisie de mots courts. Les histogrammes bleus représentent les temps de saisie moyen obtenus avec chaque technique proposée, alors que les histogrammes rouges, jaunes et verts présentent respectivement les temps de saisie pour les mots de 2, 3 et 4 lettres.

Nous pouvons observer que les deux techniques que nous proposons permettent de saisir en moyenne plus rapidement les mots courts que notre technique contrôle : les participants ont mis en moyenne $2,85$ et $2,99$ secondes avec respectivement la validation « *à frappe double* » et la validation « *à deux doigts* », alors qu'il leur a fallu $4,63$ secondes en moyenne pour saisir un mot court avec la méthode du « *glisser-taper* ». Cette différence avec la technique contrôle est significative ($W = 2361$ entre « *à deux doigts* » et « *glisser-taper* », et $W = 5562$ entre « *frappe double* » et « *glisser-taper* », avec $p < \alpha$ dans les deux cas). En revanche, il n'y a pas de différence significative entre les deux techniques que nous proposons.

D'autre part, nous pouvons voir que nos deux techniques ne sont pas dépendantes de la longueur du mot, à l'inverse de la technique « *glisser-taper* » dont le temps nécessaire pour saisir un mot augmente avec la taille de celui-ci (cette différence est significative $p < \alpha$). Ceci s'explique par le fait qu'avec la technique « *glisser-taper* », chaque caractère est saisi directement sur le clavier, et donc cette action nécessite à chaque fois du temps à l'utilisateur. Dans le cas de nos deux techniques, les mots

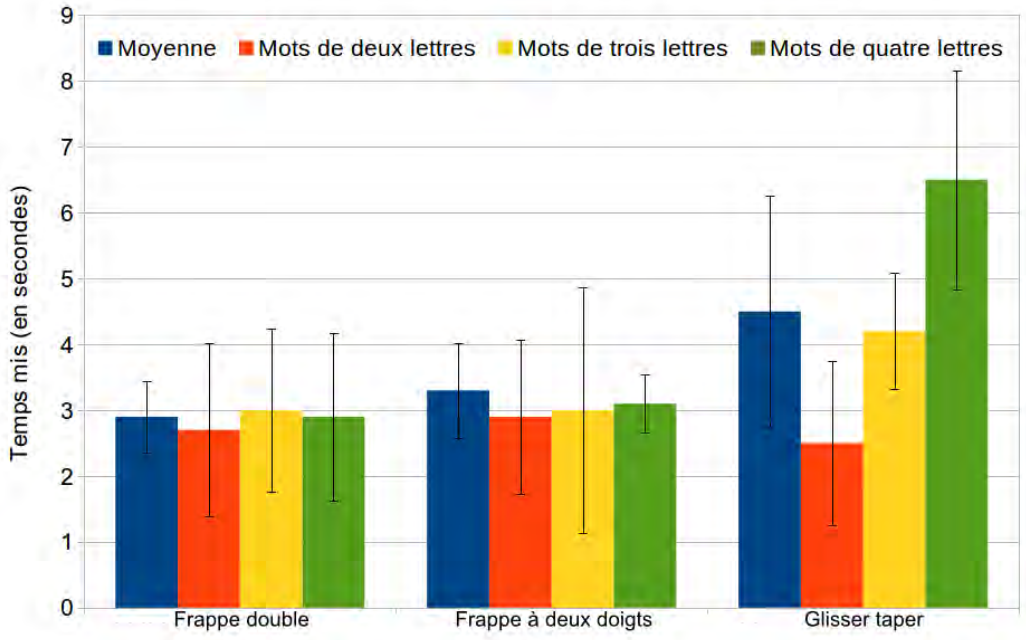

FIGURE 33 – Temps mis pour taper un mot, en fonction de la technique et de la longueur du mot

sont affichés dans une liste, et le temps nécessaire pour pointer un mot dans la liste est identique quelle que soit la longueur du mot.

2.2 Taux d'erreurs

Nous avons calculé le taux d'erreurs pour chaque technique. Cela correspond au pourcentage de mots mal sélectionnés ou saisis de manière incorrecte par rapport au nombre total de mots.

La figure 34 présente les taux d'erreurs des trois techniques. Les participants ont réalisé moins d'erreurs avec les deux techniques basées sur les listes (12% pour la validation « *à frappe double* » et 11% pour la validation « *à deux doigts* ») qu'avec la méthode « *glisser-taper* » avec laquelle les participants ont effectué en moyenne 21% d'erreurs. Ces différences sont significatives ($W = 3652$ entre « *à deux doigts* » et « *glisser-taper* », et $W = 4952$ entre « *à frappe double* » et « *glisser-taper* »,

avec $p < \alpha$ dans les deux cas).

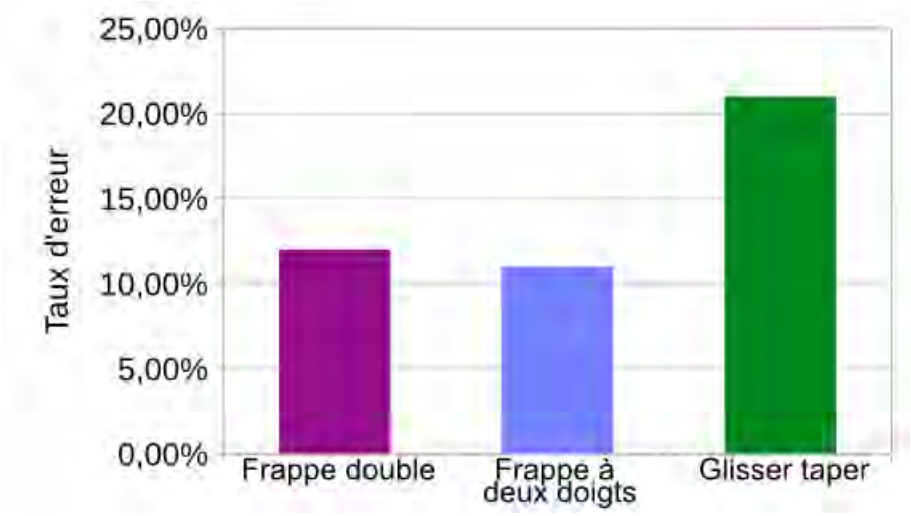

FIGURE 34 – Taux d'erreurs à la saisie d'un mot en fonction de la technique et de la longueur du mot

L'analyse des erreurs commises avec les deux techniques montre que ce n'est pas la sélection du premier caractère qui a causé ces erreurs. Aucune des erreurs ne s'est produite lors de la sélection du premier caractère du mot. Toutes les erreurs sont dues à une mauvaise sélection du mot dans la liste de mots proposés. Dans le cas de toutes les erreurs commises lors de cette expérimentation, les participants ont sélectionné le mot voisin dans la liste du mot à copier.

Bien que ce taux d'erreurs soit encore trop élevé, nos techniques à base de liste permettent quand-même de réduire le nombre d'erreurs par rapport au système DUCK où le taux d'erreurs sur les mots courts était de 20%.

2.3 Retour utilisateur

Nous avons calculé le score SUS obtenu pour chaque technique en fonction des réponses des utilisateurs aux tests. Les techniques « *glisser-taper* » et « *à frappe double* » ont obtenu un score quasiment identique avec respectivement 86 et 85. La technique « *à deux doigts* » est la troisième en terme de score avec 79.

Le classement des techniques par ordre de préférence des utilisateurs donne la technique « *à frappe double* » comme technique préférée des utilisateurs avec notamment cinq d'entre eux qui l'ont placée en première position et six qui l'ont placée en deuxième position. La technique « *à deux doigts* » a elle été classée première et deuxième par respectivement quatre et cinq utilisateurs. Enfin la technique « *glisser-taper* » est la moins appréciée des trois techniques. Elle a été classée en troisième position à huit reprises.

3 Discussion

Les résultats obtenus lors de l'expérimentation confirment que la saisie de mots courts à l'aide d'une liste est efficace. Comme le montre les résultats, les utilisateurs sont plus rapides pour saisir un mot court avec les techniques à base de liste qu'avec la méthode de « *glisser-taper* ». Cependant, ces résultats sont à relativiser car l'expérience ne se focalisait que sur la saisie de mots courts. Donc les participants n'avaient pas à réfléchir pendant leur saisie s'ils pouvaient utiliser ou non cette technique au cours de leur saisie. Lors de la saisie d'un texte avec le système DUCK, les utilisateurs auront à saisir les mots avec le système DUCK et pourront accélérer leur saisie pour les mots courts en utilisant ces listes. Cependant, il leur faudra penser à passer d'un mode de saisie à l'autre tout au long de la saisie, ce qui pourrait leur faire perdre un peu de temps. Dans l'expérience, seuls des mots courts étaient saisis donc ils n'avaient pas à se poser la question du mode de saisie et pouvaient utiliser intégralement les listes de mots courts.

D'autre part, tous les mots courts proposés font partie des mots les plus fréquemment utilisés dans la langue française. Ils étaient donc tous présents dans les listes de mots proposées. En revanche, lorsque les utilisateurs saisiront du texte avec le système DUCK, ils pourraient avoir à saisir des mots courts qui ne font pas partie des mots les plus fréquents et donc absents des listes de mots proposés. Toutefois, avec un maximum de 8 mots commençant par la même lettre, notre système permet d'utiliser 122 mots courts, soit 74% de ceux utilisés dans la langue française.

Cependant, on peut supposer qu'en utilisant ce système couramment, les utilisateurs vont apprendre petit à petit les mots se trouvant à l'intérieur des listes de mots courts. Par conséquent, avec de l'expérience, ils sauront si le mot qu'ils doivent saisir se trouve ou non dans les listes de mots proposées. Ils pourront alors utiliser plus rapidement les listes de mots courts et ainsi accéderont plus rapidement au mot voulu.

Enfin, les listes apportent aussi une plus grande précision aux utilisateurs, avec un taux d'erreurs plus faible qu'avec la technique de « *glisser-taper* » ou la saisie classique. La précision s'explique notamment par le fait que les listes proposent le mot à sélectionner sans erreur d'orthographe. De ce fait, la seule erreur possible est de mal sélectionner le mot et de prendre un mot complètement différent ; tandis que de telles erreurs sont possibles pour chaque lettre si l'utilisateur rentre le mot lettre-par-lettre (en utilisant la technique « *glisser-taper* »).

Chapitre VI

Découverte et apprentissage de DUCK pour un usage journalier

Pour conclure nos travaux sur l'optimisation des systèmes de saisie de texte sur dispositif mobile pour non-voyants, nous avons réalisé une dernière expérimentation avec notre système DUCK. Notre système intègre les améliorations proposées lors des deux chapitres précédents sur les listes de mots d'une part, et pour la saisie des mots courts d'autre part.

Nous avons choisi pour cette dernière expérimentation d'évaluer les performances de notre système à la fois par des utilisateurs novices, mais aussi après une phase de familiarisation avec notre système.

Après avoir présenté les améliorations apportées à notre système par rapport à la première étude, nous présenterons les résultats obtenus avec ce nouveau système lors des deux sessions réalisées par les participants.

1 Améliorations apportées au système DUCK

1.1 Intégration du mode mots courts

Pour l'implémentation du mode mots courts, nous avons repris l'interaction proposée dans le chapitre précédent. L'utilisateur a désormais le choix de saisir un mot en intégralité, ou, dans le cas où ce mot est un mot de la langue française de quatre lettres ou moins, il peut actionner le mode mots courts. Les interactions proposées pour utiliser le mode mots courts permettent de ne pas changer la manière de saisir des mots avec DUCK. Ainsi, dans tous les cas, l'utilisateur saisit le premier caractère de son mot de manière précise, puis, soit il continue par des frappes approximatives et une validation du mot au moyen de la liste de déduction, soit il demande à accéder à la liste des mots courts associée à ce premier caractère en effectuant à nouveau un appui sur le clavier avec deux doigts (double-press).

1.2 Saisie glissée

Nous avons également ajouté un mode supplémentaire, appelé « *saisie glisser-taper* ». Ce mode reprend le principe de la troisième technique que nous avons utilisée pour comparer les différentes interactions du mode mots courts (voir chapitre V).

Ce mode de saisie permet à l'utilisateur d'entrer, s'il le souhaite, des mots absents du dictionnaire, ou qu'il n'arrive pas à obtenir en utilisant le système de déduction. La technique revient à saisir chaque caractère un à un de manière précise : une fois que l'utilisateur a sélectionné son caractère, il le valide en appuyant un second doigt sur l'écran. Cette interaction (dual-tap) permet de distinguer la saisie de caractères indépendamment de la saisie de mots avec le système DUCK. En effet, il n'était pas possible de garder la validation des caractères en relevant le doigt de l'écran car cette interaction est déjà utilisée pour saisir le premier caractère d'un mot lors de la saisie avec DUCK. Ainsi le système s'attend après cette interaction à avoir un ensemble de frappes imprécises.

1.3 Interaction avec les listes

Enfin, pour l'ensemble des listes utilisées dans le système DUCK (que ce soit les listes de déduction de mots ou les listes statiques de mots courts), nous avons repris le choix présenté dans le chapitre III. La liste présente les mots sous forme linéaire. La sélection d'un mot s'effectue en glissant le doigt dessus, et le relâchement du doigt de l'écran valide cette sélection. Enfin la synthèse vocale énonce le mot qui est survolé par le doigt de l'utilisateur, et annonce le mot sélectionné au moment de la validation finale.

2 Protocole

2.1 Hypothèses

Lors de la première étude, nous avions constaté que le système DUCK permettait à l'utilisateur d'obtenir des performances de saisie de texte quasiment identiques à celles obtenues avec le clavier VODKA. Même si le système DUCK ne demande pas d'effort important à l'utilisateur pour que celui-ci puisse le prendre en main, nous pensons qu'avec de l'entraînement, un utilisateur aura de meilleures performances avec DUCK qu'avec le clavier VODKA. C'est pourquoi, cette étude est constituée de deux sessions séparées d'une phase d'entraînement de deux semaines. Notre principale hypothèse est qu'après cette phase d'entraînement, les participants auront de meilleures performances avec DUCK qu'avec VODKA.

2.2 Participants

4 personnes non-voyantes, 2 femmes et 2 hommes, âgées de 45 ans en moyenne, ont participé à cette étude. Ces quatre participants travaillent ou sont élèves dans un Centre d'Education Spécialisé pour Déficients Visuels (CESDV) dont l'IRIT est partenaire. Le Centre a mis un local à notre disposition pour la durée de l'expérimentation. Les participants sont des utilisateurs réguliers d'un smartphone et ont l'habitude d'utiliser un clavier logiciel accompagné de VoiceOver. Aucun des partici-

pants n'avait participé à la première expérimentation, ni aux études sur les listes ou mots courts.

2.3 Matériel

Les participants ont utilisé le même smartphone que dans la première étude (un Samsung Galaxy SIII), avec les mêmes caractéristiques et outils logiciels utilisés lors de la première expérience.

De la même manière que pour la première expérience, deux claviers étaient présentés aux participants : d'une part le système DUCK avec les améliorations apportées suite aux études précédemment décrites ; et d'autre part, le clavier VODKA tel qu'il était déjà utilisé lors de la première expérimentation. La disposition des touches et caractères est aussi identique à la première expérimentation.

2.4 Procédure

Lors de chaque exercice des deux sessions, la tâche demandée à l'utilisateur était de taper une série de phrases le plus rapidement possible. Avant de commencer la première session, chaque participant pouvait se familiariser avec le système à tester. Cette période dite de familiarisation permet à l'expérimentateur d'expliquer au sujet le fonctionnement du clavier. Le participant apprend à utiliser le clavier, à taper un ou plusieurs mots, puis une ou plusieurs phrases, selon son souhait.

Lorsque le participant se sentait suffisamment à l'aise, il devait alors saisir des phrases pendant 20 minutes, ou jusqu'à ce que ce dernier ait saisi 20 phrases. Il était demandé au sujet de ne pas s'arrêter entre les phrases. La tâche de saisie d'une phrase était simple : la phrase à saisir était lue au participant par synthèse vocale. Il devait ensuite taper cette phrase le plus rapidement possible. Si l'utilisateur le souhaitait, il pouvait réécouter à sa guise la phrase à saisir.

Le système vérifiait à la fin de la saisie de chaque mot si celui-ci correspondait ou non au mot que devait saisir l'utilisateur. Si le mot était erroné, l'utilisateur devait alors ressaisir le mot en question, jusqu'à ce que celui-ci soit correct. Afin

de simplifier la tâche pour les participants, nous n'avons pas demandé à ce que ces derniers saisissent de façon exacte chaque mot : une tolérance "homophonique" de chaque mot suffisait. Ainsi, si l'utilisateur devait saisir "sait", le système considérait comme correctes les entrées homophones, par exemple "ses", "ces" ou encore "sais".

Une fois la phase de saisie terminée, le sujet devait remplir un questionnaire dans lequel il indique son avis sur le clavier qu'il vient d'utiliser.

2.5 Conception

Afin de mesurer l'évolution des performances des participants au cours des premières utilisations du système DUCK, nous avons réalisé deux sessions d'évaluation. Les deux sessions ont été séparées de deux semaines pendant lesquelles les participants pouvaient utiliser à leur guise le dispositif.

La première session s'est déroulée dans une pièce isolée en présence de l'expérimentateur, et ce pour une durée variant de 1h à 2h suivant les performances du participant. Le participant était assis et pouvait poser le dispositif sur un bureau afin de réaliser l'expérience dans des conditions optimales.

La session était composée de deux exercices : un pour chaque système à tester (DUCK et VODKA). Deux participants ont commencé par l'exercice avec le système DUCK, avant de réaliser celui avec le clavier VODKA. Les deux autres participants ont réalisé les exercices dans l'ordre inverse.

Après la première session, le dispositif de saisie a été laissé en libre accès aux participants pendant deux semaines, par l'intermédiaire du CESDV. Il leur était demandé de venir utiliser le clavier quotidiennement, ou à défaut le plus souvent possible, pour réaliser une production écrite correspondant à quelques lignes de texte, à l'image d'un email. Le dispositif expérimental était réglé pour afficher le clavier automatiquement au plus deux minutes après l'allumage. Il était demandé aux utilisateurs de rendre compte de leur apprentissage (date et heure de début, durée de l'apprentissage). Dans notre cas, le personnel du CESDV s'est occupé de noter les noms et heures de début et de fin des sessions journalières. Pour effectuer leur saisie, les utilisateurs avaient accès à une pièce calme, où il pouvait s'asseoir et utiliser le dispositif sur un bureau.

A la fin de ces deux semaines, nous avons réalisé une seconde session dont la procédure était identique à la première session.

2.6 Données collectées

Lors de chaque exercice, nous avons enregistré l'ensemble des actions effectuées par l'utilisateur (pression, relâchement, et déplacements du doigt à l'écran), les instructions données par le système à l'utilisateur (consigne de l'exercice, phrases à recopier) ainsi que les listes de mots présentées à l'utilisateur au cours de la saisie (celles produites par le système de déduction ainsi que celles utilisées pour la gestion des mots courts). Enfin, nous avons aussi recueilli les réponses obtenues au questionnaire SUS ; ainsi que ses retours informels formulés par les participants à la fin de chaque session.

3 Résultats

Nous détaillons ici les résultats obtenus lors des deux sessions de notre étude. Dans un premier temps, nous présentons les résultats quantitatifs. Nous avons réalisé des tests de Wilcoxon ainsi que des analyses de Friedmann pour appuyer nos résultats dans le cas où ces derniers ne suivent pas une loi normale. Le seuil de confiance α a été fixé à 0,05.

3.1 Vitesse de saisie de texte

De façon similaire à la première étude, pour analyser la vitesse de saisie des deux systèmes, nous nous sommes focalisés sur les mots qui ont été tapés correctement.

3.1.1 Évolution de la saisie au cours des deux sessions

Nous nous sommes ici intéressés aux résultats des deux sessions de l'étude. Dans un premier temps, nous avons choisi de ne considérer que la première session, sans

entraînement, puis la seconde session, après la phase d'entraînement de deux semaines des utilisateurs.

Le graphique présenté en figure 35 nous donne les différentes vitesses de saisie par clavier pour chaque session. Les vitesses moyennes d'entrée de chaque clavier sont pour la première session de 0,71 cps ($\sigma = 0,49$, médiane = 0,65) pour DUCK contre 0,56 cps ($\sigma = 0,12$, médiane = 0,57) pour VODKA. Cette différence de vitesse entre les deux claviers est significative ($W = 1113$, $p = 0,0478$). A la seconde session, les participants ont obtenu une vitesse de saisie de 1,1 cps ($\sigma = 0,68$, médiane = 1,1) avec le système DUCK, alors qu'ils n'ont saisi en moyenne que 0,61 cps ($\sigma = 0,13$, médiane = 0,59) avec le clavier VODKA. Cette différence est significative ($W = 859$, $p = 0,0364$). Ces résultats confirment donc notre hypothèse d'obtenir une meilleure vitesse de saisie avec DUCK qu'avec VODKA.

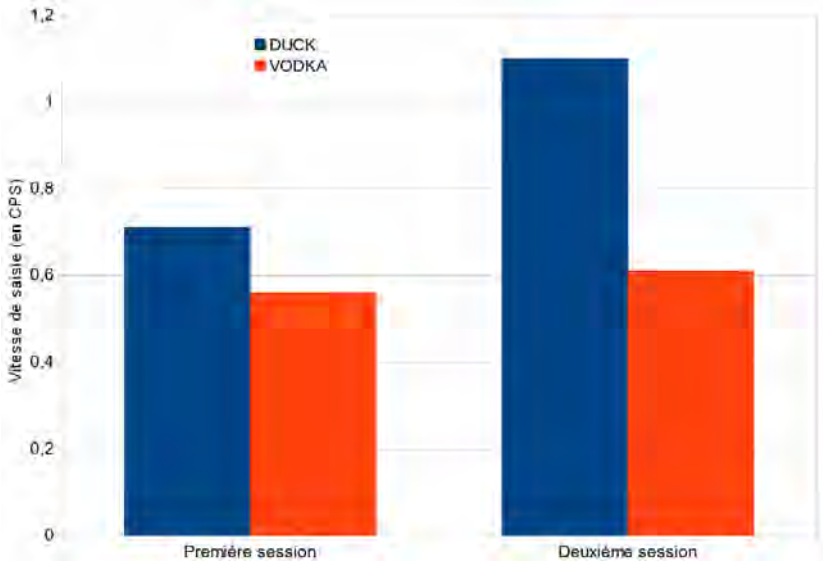

FIGURE 35 – Vitesse de saisie des deux claviers à l'issue de la première et de la deuxième session

D'autre part, on peut constater une nette évolution de la vitesse de saisie avec le système DUCK entre les deux sessions. La différence de vitesse de saisie entre les deux sessions est aussi significative pour chacun des systèmes testés (DUCK : $W = 649$, $p = 0,024$ et VODKA : $W = 1835$, $p = 0,097$).

3.1.2 Vitesses de saisie et longueur de mot

Les figures 36 et 37 présentent la vitesse de saisie moyenne d'un mot en fonction de la longueur de ce mot respectivement à l'issue de la première session expérimentale, et lors de la seconde session.

Sur la figure 36, nous pouvons constater que DUCK est plus efficace que VODKA pour les mots de quatre caractères ou plus. D'autre part, la vitesse de saisie est plus élevée pour les mots long qu'elle ne l'était pour ce même type de mots dans la première expérimentation. En analysant le temps de saisie avec DUCK, nous pouvons constater que le temps nécessaire à la validation d'un mot dans la liste de déduction n'est que de 2,85 secondes en moyenne alors qu'il était de 3,6 secondes lors de la première étude. Ceci vient donc confirmer que les modifications apportées à l'interaction avec la liste sont bénéfiques sur le temps de validation, et permettent ainsi à l'utilisateur de saisir plus rapidement.

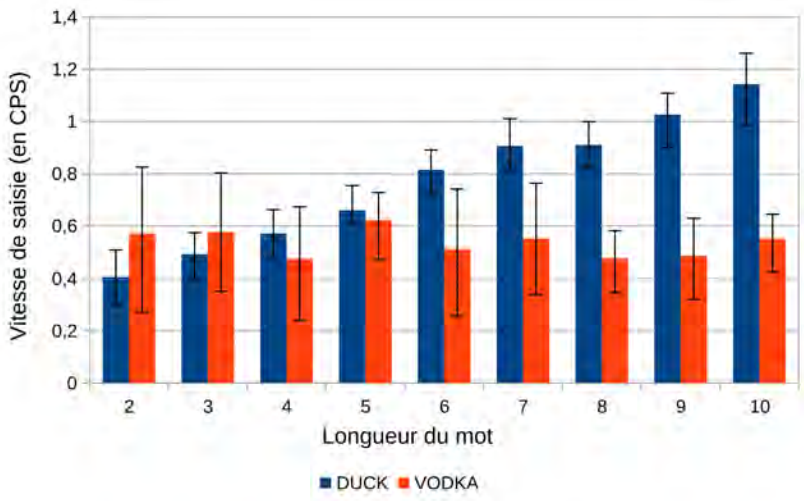

FIGURE 36 – Vitesse de saisie des deux claviers à l'issue de la première session

Après la phase d'apprentissage, lors de la deuxième session, les participants ont amélioré leur vitesse de saisie de texte avec le système DUCK, et notamment pour les mots courts (voir figure 37). Il n'y a plus de différences significatives sur la saisie des mots courts (mots constitués de quatre caractères ou moins) entre les deux systèmes.

Ceci peut s'expliquer par la bonne utilisation par les participants des listes de mots courts. D'autre part, la décomposition du temps de saisie, avec le clavier DUCK, entre la phase de frappes et la phase de validation montre que les utilisateurs ont mis en moyenne 2,07 secondes lors de la phase de validation. Ainsi, avec la phase d'apprentissage, les participants ont réduit le temps nécessaire pour valider un mot dans la liste de déduction.

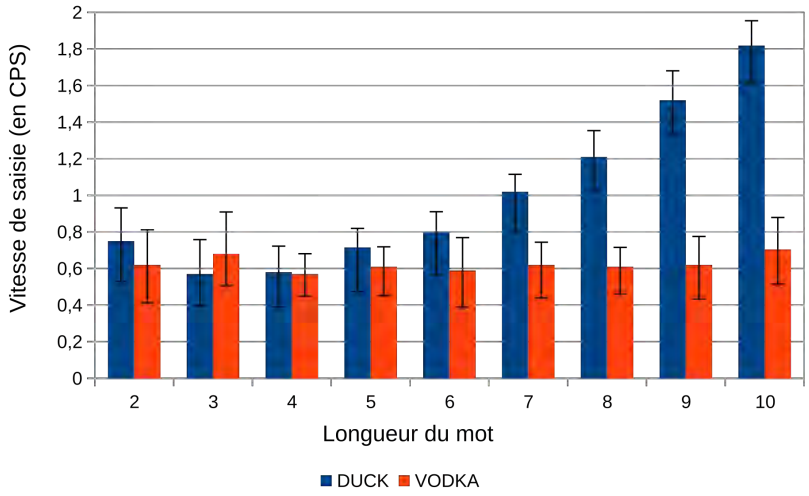

FIGURE 37 – Vitesse de saisie des deux claviers à l'issue de la deuxième session

Enfin, la différence de vitesse s'accentue avec la longueur du mot : l'utilisateur est deux fois et demi plus rapide avec DUCK qu'avec VODKA pour des mots de dix lettres (1,81 cps pour DUCK contre 0,70 cps pour VODKA).

3.1.3 Temps de pause intercaractère

Enfin, concernant le temps de saisie, nous avons analysé le temps de pause intercaractère. Ce temps correspond au temps "perdu" par l'utilisateur lors de la saisie : une saisie idéale correspondrait en effet à un enchaînement de caractères quasi-constant. Selon Kinkead [Kinkead, 1975], ce temps est minimisé à un tiers de seconde pour un clavier physique, et tout temps passé entre deux caractères est compté comme "perdu".

Pour mesurer le temps de pause intercaractère, nous avons calculé la différence temporelle entre le moment où l'utilisateur relâche son doigt du clavier pour saisir le caractère n et le moment où il appuie sur le clavier pour commencer la saisie du caractère $n + 1$.

La figure 38 présente les temps intercaractères mesurés pour chaque clavier. Pour la première session, ce temps est de 1,124 secondes pour DUCK ($\sigma = 0,36$) et de 1,78 secondes pour VODKA ($\sigma = 0,4$). Pour la deuxième session, ce temps n'est que de 0,658 secondes pour DUCK ($\sigma = 0,36$) et 1,75 secondes pour VODKA ($\sigma = 0,48$). Dans le cas de VODKA, il n'y a pas de diminution significative du temps de pause intercaractère ($p = 0,256$). En revanche, pour le système DUCK, la diminution du temps de pause intercaractère est significative entre les deux sessions ($W = 986$ $p = 0.0126$). Cette différence traduit la confiance que l'utilisateur prend au fur et à mesure de l'utilisation du système. L'utilisateur n'hésite plus à frapper approximativement sur le clavier, et par conséquent effectue ses frappes de plus en plus rapidement.

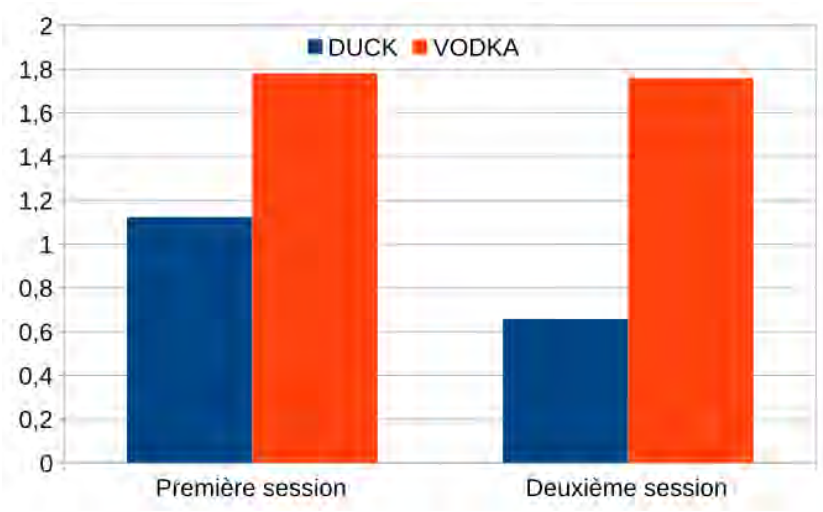

FIGURE 38 – Évolution du temps de pause intercaractère

3.2 Précision des claviers

3.2.1 Taux de réussite

Pour comparer la précision et l'efficacité de chaque clavier, il a été nécessaire d'introduire la notion de « *mot correct* ».

Un mot est défini comme étant un « *mot correct* » si, et seulement si, le mot correspond phonétiquement au mot demandé dans la phrase.

Ainsi, pour le mot "veut", les mots "veux" et "veut" sont considérés comme corrects. Nous avons effectué ce choix pour éviter de pénaliser les utilisateurs sur l'orthographe. Le temps de saisie a été mesuré par rapport au mot effectivement tapé pour éviter de fausser les résultats avec des homophones de longueur différente.

Pour pouvoir comparer les deux claviers, nous avons calculé un taux de réussite, T_R, calculé simplement par la formule

$$T_R = \frac{nombre\,de\,mots\,corrects}{nombre\,de\,mots\,saisis}$$

Pour les deux claviers, nous obtenons les résultats présentés sur la figure 39 : pour la première session, DUCK affiche un taux de réussite de 75% ($\sigma = 5,9\%$) contre 52% ($\sigma = 3,2\%$) pour VODKA. Dans le cas de la deuxième session, le taux de réussite indique 91% ($\sigma = 8,3\%$) pour DUCK contre 77% ($\sigma = 6,8\%$) pour VODKA.

Un test de Shapiro-Wilk indique que les distributions présentées ne suivent pas une loi normale (p <0,01). Nous avons alors réalisé deux tests de Wilcoxon pour comparer les résultats obtenus. Le premier test indique que DUCK est plus précis que VODKA quant à la saisie de mots corrects pour la première session ($W = 1070$, $p = 0.036$), tandis que le deuxième test confirme cette tendance pour la deuxième session ($W = 2069$, $p = 0.032$).

D'autre part, nous pouvons aussi observer que pour les deux systèmes, le taux de réussite augmente entre les deux sessions. Cette tendance est confirmée par des tests de Wilcoxon ($W = 1208$, $p = 0.0487$ pour DUCK et $W = 276$, $p = 0.0124$ pour VODKA).

Au vu de ces observations, nous pouvons en conclure que DUCK est plus efficace

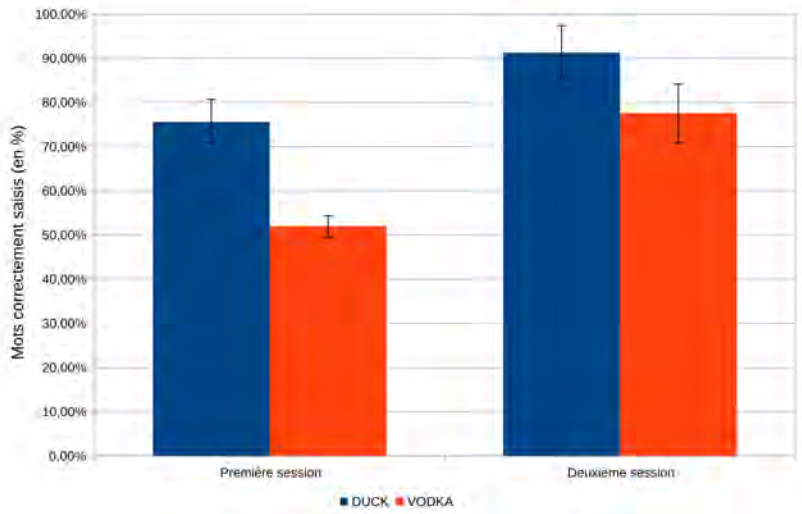

FIGURE 39 – Réussite par clavier

que VODKA au niveau de la correction des mots : le nombre de mots correctement saisis est plus important avec DUCK qu'avec VODKA.

3.2.2 Utilisation de la fonction mots courts

Nous nous sommes enfin intéressés à l'utilisation de la fonction mots courts pour la saisie de mots au long de l'expérience.

Notre première hypothèse était de considérer que l'activation du mode mots courts permettait de rendre la saisie plus efficace.

Afin de mesurer l'impact de l'utilisation de la fonction mots courts, nous avons comptabilisé le nombre d'utilisations de la fonction mots courts, que nous avons séparée en deux types d'utilisation :

1. Les utilisations "utiles", c'est-à-dire les cas où l'utilisateur doit véritablement taper un mot court ;
2. Les utilisations "inutiles", c'est-à-dire les cas où l'utilisateur active le mode par erreur.

Dans le cas de l'utilisation inutile, nous ne distinguons pas les cas où l'utilisateur

a activé consciemment le mode mots courts par erreur (par exemple, en supposant que le mot à saisir s'y trouvait alors qu'il n'y était pas) du cas où il s'agissait d'un mouvement involontaire ; dans les deux cas, l'activation du mode mot court consiste pour nous en une baisse de performance (temporelle et actions), et donc en un critère négatif.

Pour évaluer l'utilisation de la fonction « *mots courts* », nous avons produit un score représentatif. Par exemple, sur une phrase contenant deux mots courts ("la maison est rouge"), où les mots "la" et "est" sont deux mots courts qu'il est possible de saisir par ce mode, le nombre d'utilisations idéal est de 2 (une fois pour "la" et une fois pour "est"). Avec ce nombre d'utilisations idéal, nous pouvons calculer le taux d'utilisation de la fonction « *mots courts* » $Taux_{MC}$ comme le nombre d'utilisations réellement utiles du mode mots courts effectuées par l'utilisateur divisée par le nombre d'utilisations idéal. Nous obtenons ainsi :

$$Taux_{MC} = \frac{\sum_{i \in Mots} \sigma(i)}{N_{ideal}}$$

où $\sigma(i)$ renvoie 1 si, et seulement si, le mot i est un mot court et l'utilisateur a invoqué le mode mots courts. N_{ideal} correspond au nombre de fois où l'appel est utile (c'est à dire au nombre de mots courts à saisir par l'utilisateur).

De même, nous avons proposé également de calculer le taux d'appels incorrects du mode mots courts par :

$$TauxIncorrect_{MC} = \frac{\sum_{i \in Mots} \beta(i)}{N_{noncourts}}$$

où $\beta(i)$ correspond à une fonction qui renvoie 1 si, et seulement si, le mot i n'est pas un mot appartenant à l'ensemble des mots courts proposé et l'utilisateur a invoqué le mode mots courts. $N_{noncourts}$ correspond au nombre de mots saisis par l'utilisateur et absents des listes de mots courts.

Nous faisons l'hypothèse que l'utilisateur devient plus précis dans ses gestes et qu'il connaît mieux les listes de mots au fur et à mesure de sa saisie. Ainsi, nous devrions observer une augmentation du taux d'utilisation de la fonction des mots

courts utiles, et une diminution des utilisations incorrectes.

Nous présentons les résultats que nous avons obtenus dans la figure 40. Le taux d'utilisation correct de la fonction mots courts s'élève à 43% pour la première session, contre 68% pour la seconde session. Un test de Wilcoxon confirme la différence significative de l'utilisation correcte du mode mots courts ($W = 511$, ($p<0,036$)). Cette tendance s'explique par une meilleure connaissance par les utilisateurs du contenu des listes de mots courts lors de la seconde session.

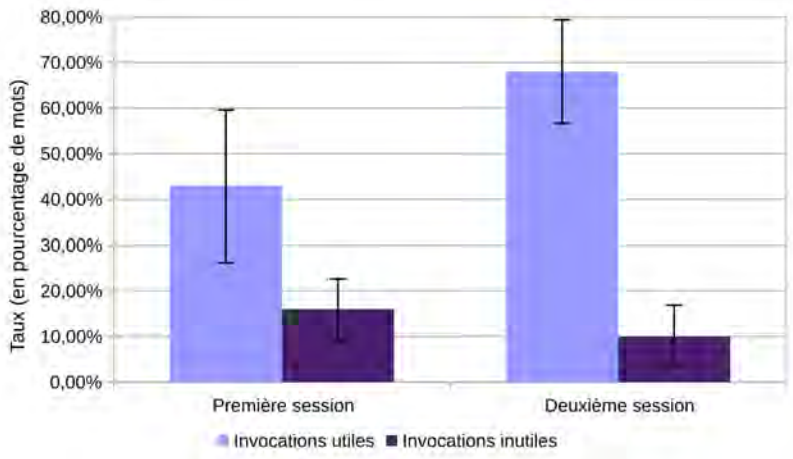

FIGURE 40 – Utilisation de la fonction mots courts

Concernant le taux d'appels inutiles à la liste, les participants en ont réalisé 16% lors de la première session et 10% lors de la seconde session. Cependant, cette diminution n'est pas significative ($W = 421$, ($p<0,280$)). D'autre part, les participants ont expliqué à la fin de la seconde session que les appels inutiles au mode mots courts étaient en grande partie dû à une interaction erronée, le plus souvent à cause d'un glissement de doigt intempestif.

3.3 Avis des utilisateurs

Les retours de nos utilisateurs ont grandement évolué au cours de l'expérience. Initialement, sur nos quatre utilisateurs, l'avis donné à DUCK était mitigé (seulement une personne sur quatre pensait envisager de l'utiliser de façon quotidienne) jugeant

le clavier certes intuitif mais son utilisation perturbante : « *je n'aime pas ne pas avoir de retour lorsque je saisis* ». Deux sujets sur quatre ont affirmé avoir eu du mal à tester le clavier de façon quotidienne pour s'y entraîner, préférant toujours la saisie par reconnaissance vocale pour leur quotidien.

Nous avons présenté les scores sur la figure 41. Nous observons qu'initialement, DUCK est moins apprécié que VODKA (82 contre 87) ; mais la tendance s'inverse une fois l'entraînement terminé (DUCK reçoit le score de 89 tandis que VODKA descend à 77).

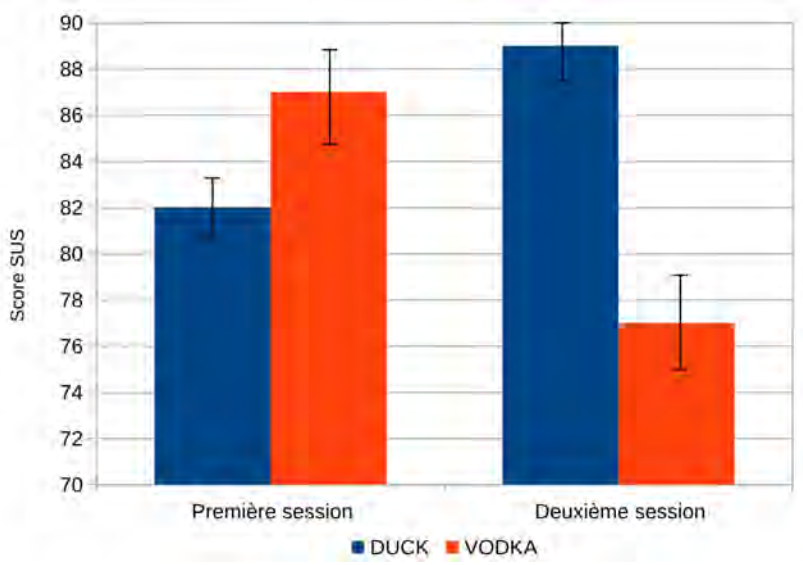

FIGURE 41 – Évolution des scores SUS pour les deux claviers

Les remarques des utilisateurs sont différentes. Trois utilisateurs sur quatre indiquent que le clavier est « *addictif* » : « *à la suite de mes saisies d'entraînement sur le prototype, je trouvais surprenant que le clavier par défaut de mon téléphone ne soit pas aussi tolérant* ». Un utilisateur indique que le clavier DUCK lui a permis de « *franchir le pas* » et de se mettre non plus à la saisie uniquement vocale, mais à la saisie sur clavier logiciel sur son téléphone : « *je ne savais pas que je connaissais le clavier aussi précisément, je me sens plus à même d'utiliser moins souvent la reconnaissance vocale maintenant* ». À la sortie de l'expérience, seul un sujet sur quatre

affirme ne pas vouloir utiliser le clavier DUCK de façon quotidienne : « *j'ai bien compris comment le clavier fonctionne, mais je n'aime pas la façon dont il me fait saisir du texte* ». Les commentaires négatifs de la part des utilisateurs ciblent surtout deux aspects "personnels" du clavier : « *je n'aime pas la voix utilisée* », « *il faudrait pouvoir ajouter des mots, des symboles et d'autres types de saisie* », et « *les interactions proposées sont efficaces, mais même avec de l'entraînement, j'active parfois le mode mots courts par erreur : il faudrait pouvoir changer les interactions suivant nos préférences* ».

4 Discussion

Cette dernière expérimentation avait pour but d'évaluer le système DUCK intégrant les améliorations apportées sur la gestion des mots courts et sur la validation des mots dans les listes de déduction. Nous avions choisi de réaliser cette expérimentation en deux sessions séparées par une phase d'apprentissage. Cela permet de voir les résultats par des utilisateurs totalement novices, puis une fois qu'ils ont pris en main le système pendant deux semaines.

De manière globale, nous pouvons constater que les améliorations apportées à DUCK ont été bénéfiques. En effet, le système est plus performant que VODKA dès la première utilisation. Les participants ont ainsi obtenu une vitesse de saisie significativement plus élevée avec DUCK qu'avec VODKA dès la première session. De même, ils ont effectué moins d'erreurs de saisie avec DUCK.

Au travers de cette évaluation, nous avons pu observer l'amélioration des performances concernant la validation du mot dans la liste de déduction. En effet, lors de la première évaluation, ce temps était de 3,5 secondes pour valider un mot. Lors de la première session de cette expérimentation, le temps de validation moyen n'était plus que de 2,85 secondes. Ce temps était encore réduit lors de la seconde session avec seulement 2,07 secondes nécessaires pour valider un mot dans la liste de déduction. Cette réduction du temps de validation a ainsi permis d'augmenter la vitesse de saisie des utilisateurs avec DUCK. Les participants ont ainsi des vitesses de saisie

plus élevées avec DUCK pour des mots de 4 caractères ou plus lors de la première session, contre 6 caractères ou plus lors de la première expérimentation.

D'autre part, les participants ont aussi bien assimilé le mode mots courts. S'ils ne l'ont utilisé que dans 40% des cas lors de la première session, ils s'en sont servis dans deux tiers des cas lors de la seconde session. Cette bonne utilisation du mode mots courts permet ainsi aux utilisateurs de saisir aussi rapidement, voire plus rapidement avec DUCK, pour tous les mots quelle que soit leur longueur.

Enfin, nous pouvons constater qu'après une phase d'apprentissage, les participants améliorent de manière significative leur performance de saisie avec le clavier DUCK. La vitesse et la précision lors de la saisie ont été améliorées. Avec une utilisation plus longue, les participants trouvent ainsi des avantages à utiliser ce système plutôt que ceux qu'ils utilisaient déjà.

Conclusion

La saisie de texte sur dispositif mobile au moyen d'une approche logicielle est toujours à double tranchant : elle offre de grands avantages en termes de dynamisme pour le concepteur et pour l'utilisateur, malgré une grande insuffisance pour les utilisateurs en situation de déficience, et particulièrement dans le cas de la déficience visuelle. Il existe aujourd'hui deux principales approches pour pallier cela : soit on fournit un système adapté à l'utilisateur, quitte à imposer un temps d'apprentissage conséquent ; soit on propose une interaction ou un alphabet différent, plus adapté, réduisant ainsi le temps d'apprentissage nécessaire, au détriment d'un accès complet à la saisie de texte. Dans les deux cas, nous jugeons qu'une approche mixte, offrant un système adapté mais utilisant une interaction similaire et un appui sur une connaissance existante du clavier, est plus efficace. Pour cela, nous avons travaillé selon trois points :

— Tout d'abord, il est nécessaire de penser un système qui permette à l'utilisateur de saisir un mot de façon efficace ; si possible en éliminant un maximum de temps morts. Pour répondre à cette contrainte, il nous a paru le plus pertinent de proposer une saisie approximative qui sera corrigée (déduite) par la suite par le système, en offrant une interaction adaptée aux besoins de l'utilisateur : ainsi, en évitant de surcharger l'environnement auditif de l'utilisateur, nous arrivons à offrir une saisie plus efficace en termes de rapidité et de précision qu'une saisie classique ; et ce, sans apprentissage.

— Ensuite, l'optimisation du retour de l'information à l'utilisateur : toujours dans l'optique de ne pas être dans une lacune de l'information ni dans une surcharge de l'environnement offert à l'utilisateur, nous avons déterminé l'in-

teraction adéquate pour que le déficient visuel puisse rapidement trouver le mot qu'il souhaite taper, et ce, en minimisant les risques de confusions.

— Pour terminer, nous nous sommes intéressés sur le cas charnière des mots dits « *mots courts* », mots pour lesquels le système déductif peut être contraignant et moins performant que le mode classique. Nous avons cherché une interaction qui permettrait de compenser les cas limites, sans nuire au fonctionnement actuel du système de déduction. Pour cela, nous avons conçu un mode intermédiaire, le mode mots courts, destiné à permettre la saisie de mots couramment utilisés sans perte de temps.

L'ensemble des résultats que nous avons présentés ci-dessus montre que notre objectif de fournir une méthode de saisie de texte efficace et utilisable sans apprentissage a été atteint. Cependant, nous ne pouvons pas encore maximiser l'impact du gain par rapport à un clavier logiciel classique dès le premier abord : comme nous l'avons montré dans notre dernière étude, un apprentissage améliore grandement les performances de notre clavier. Une possible solution déjà évoquée plus tôt serait de permettre une adaptation du clavier aux besoins de l'utilisateur ; voir un certain apprentissage : au fur et à mesure de la saisie, le clavier deviendrait de plus en plus efficace, et ne se limiterait plus qu'à la saisie au niveau du mot.

Ouvertures

La première possible ouverture concerne une gestion logicielle du contexte : jusqu'à présent, nous ne nous contentons d'analyser les frappes de l'utilisateur. Une meilleure prise en charge de l'utilisateur concerne l'analyse de la phrase. En effet, nous nous sommes jusqu'à présent limités uniquement à la saisie effective d'un mot. Il y a cependant de nombreux paramètres que nous avons actuellement ignorés pour rendre l'interaction simple. Cependant, comme mentionné à plusieurs reprises dans nos travaux, une analyse de la phrase au niveau grammatical ou conjugationel, par exemple, permettrait de mieux prédire les mots proposés (par exemple, si le sujet de la phrase est un "nous", le système de correction proposerait dans un premier temps une terminaison en "-ons" qu'une terminaison en "-ont"). La simple utilisation d'un modèle linguistique plus poussé (en utilisant la fréquence d'utilisation de certains mots par l'utilisateur, par exemple) et l'analyse des relations entre les mots permettrait d'augmenter grandement l'efficacité de la prédiction (une simple analyse grammaticale typée[De Marneffe et al., 2006] par exemple).

Dans un second temps, il serait intéressant de considérer un système de correction plus efficace : pour le moment, la seule correction possible concerne l'effacement du mot que l'on vient de saisir. Il n'y a aucune interaction permettant de "remonter" dans la succession des mots dans la phrase pour permettre un contrôle plus fin. De même, la saisie n'est possible qu'au niveau mot : il serait bénéfique pour la saisie de voir s'il n'est pas possible d'ajouter un "seuil de tolérance" qui permette à l'utilisateur de corriger ou d'enlever une ou plusieurs frappes accidentelles lors de la phase "saisie approximative" de l'entrée d'un mot ; ou de pouvoir revenir en arrière en utilisant le contexte de la phrase pour ajouter une possibilité de correction supplémentaire des

mots précédemment tapés.

La troisième ouverture que nous envisageons concerne la gestion de la ponctuation. Bien que notre système dans son état actuel permette de saisir des mots, les chiffres et symboles ne sont pris en compte que de façon sporadique et "traditionnelle" : par un mouvement ou un geste sur l'écran, une grille de type "exploration au toucher" apparaît et l'utilisateur peut choisir le symbole désiré. Sachant qu'un symbole de ponctuation n'est jamais seul dans une phrase, il serait intéressant de rendre l'interaction de l'insertion du caractère espace "intelligente" : ainsi, un simple geste et une étude de la forme de la phrase (par exemple, la présence d'une inversion sujet verbe souligne une question), ou une analyse du mot précédent (il y a déjà une incise ouverte dans la phrase que l'on souhaite refermer, ainsi le système propose une virgule), ou même l'utilisation d'un mode "ponctuation courte" pour remplacer l'interaction de sélection fastidieuse.

Appliquer ces différentes approches permettrait d'étendre la saisie non plus à une saisie par mot, mais à des phrases, par exemple. Ainsi, la frappe deviendrait plus fluide : l'interaction de sélection des mots courts et de validation des mots, limitante temporellement dans notre étude, serait utilisée beaucoup moins souvent en moyenne (une fois par phrase contre une fois pour chaque mot). En suivant les souhaits des utilisateurs, et en permettant la personnalisation du système, l'interaction gagnerait en fluidité et en dynamisme : pour le moment, l'interface est rigide et les interactions ne sont pas modifiables. De plus, l'interaction de type glisser-taper, qui permet à l'utilisateur d'ajouter un mot au dictionnaire et de saisir un mot inhabituel, pourrait grandement être améliorée. L'exploration de nouvelles interactions externes à la saisie est également à prendre en compte : quelle interaction peut-on proposer à l'utilisateur pour changer de langue, par exemple ? Nous pouvons influencer de nombreux paramètres pour rendre le système plus efficient sur une saisie à long terme.

Il est également possible d'influencer sur le type de dispositif utilisé : il existe des approches récentes qui proposent une technologie permettant d'afficher du Braille de façon dynamique ([Ren et al., 2008]), adaptée aux dispositifs mobiles ainsi qu'aux liseuses portatives. En ajoutant un moyen d'entrée tactile à ces derniers (par l'ajout d'une surface électrosensible, par exemple), il devient possible de rendre le retour de

DUCK beaucoup plus précis, et d'utiliser le toucher pour produire une interface plus réactive (en donnant, par exemple, une impression de la forme des touches actuelles).

De plus, les problèmes de petite taille et de précision ne sont pas limités simplement à la population déficiente visuelle. En effet, nous avons travaillé dans le contexte spécifique d'une saisie parasitée par une absence de retour, que nous avons corrigée par l'utilisation d'une approximation de l'interaction pour la simplifier. Cependant, notons que les problèmes d'exploration sont également très coûteux pour les déficients moteurs[Wobbrock and Myers, 2006]. Dans les deux situations, la saisie déductive constitue en soi une solution qui nous paraît correspondre efficacement aux problèmes de saisie : une proposition de futurs travaux serait de rendre le clavier DUCK universellement accessible, adapté à d'autres déficiences que la déficience visuelle.

Enfin, la communication écrite s'utilisant de plus en plus sur les dispositifs mobiles, et notamment dans les domaines de l'éducation[Villemonteix and Khaneboubi, 2012, Lindquist et al., 2007], on assiste plus à un véritable changement de support qu'à une simple suppléance ; il devient alors également intéressant d'améliorer de façon efficace la saisie et l'utilisation de tels dispositifs dans des domaines de saisie spécifiques (par exemple, dans le domaine des mathématiques). Un axe à développer serait de permettre à l'utilisateur d'entrer rapidement du texte correspondant à des applications particulières (par exemple, le standard MathML[Ausbrooks et al., 2003]) pour que sa saisie devienne aussi efficace que celle d'un utilisateur ne présentant aucune déficience sur des outils de saisie rapide (dans le cas d'une utilisation pour un échange d'idées[Lucero et al., 2010], par exemple). Dans de telles situations, le dispositif mobile devient bien plus qu'un simple palliatif, il est le seul moyen pour l'utilisateur en situation de handicap de pouvoir exprimer ses idées et les agencer ou communiquer avec les autres. Cette proposition permettrait d'avoir un clavier universel, qui permette une saisie plurielle, non limitée uniquement à la saisie journalière.

Appendices

Annexe A

Consignes données aux participants

1 Première étude sur le clavier DUCK

1.1 Lors de l'accueil des participants

« *Dans cette expérience, vous allez tester deux systèmes de saisie de texte expérimentaux qui ont été développés au laboratoire. Pour cela, on va vous demander d'écrire des phrases qui vous seront dictées. Vous n'êtes pas évalués au cours de ces expériences anonymes. Ce sont les systèmes de saisie qui seront comparés. On vous demandera donc votre préférence à la fin des expériences* »

1.2 Lors de la phase de familiarisation

« *Vous allez vous familiariser avec les deux systèmes de saisie de texte et les fonctions du système. Pour cela vous aurez plusieurs instructions à suivre et des phrases à recopier. Soyez tranquille, vous pouvez recommencer cette phase plusieurs fois jusqu'à vous sentir à l'aise.* »

1.2.1 Lors de la première utilisation du clavier VODKA

« *Vous allez maintenant vous familiariser avec la saisie d'un mot.*
Pour taper un mot...
Vous allez maintenant vous familiariser avec la saisie d'une phrase
Pour finir une phrase... »

1.2.2 Lors de la première utilisation du clavier DUCK

« *Vous allez maintenant vous familiariser avec la saisie d'un mot.*
Pour taper un mot...
Pour valider un mot, vous devez taper l'écran avec 2 doigts
Vous allez maintenant vous familiariser avec la saisie d'une phrase
Pour finir une phrase... »

1.3 Lors de la phase d'évaluation

« *Nous allons vous demander de recopier le plus rapidement possible, tout en faisant le moins d'erreurs possible, plusieurs phrases dictées par le système. Afin de maximiser votre vitesse de frappe, nous vous conseillons de mémoriser chaque phrase avant de commencer la saisie. Vous pouvez à tout moment réécouter l'énoncé si vous n'en êtes pas sûr en faisant glisser deux doigts vers le haut ; et écouter ce que vous avez tapé en faisant glisser deux doigts vers le bas. Il est demandé de taper uniquement les caractères alphabétiques, sans se soucier de la ponctuation, de la casse ou des majuscules.*
Vous pouvez vous arrêter à tout moment entre deux tâches de recopie, mais nous vous demandons de le faire préférentiellement lorsque vous avez terminé la saisie d'une phrase, avant de commencer la phrase suivante. Veuillez ne pas vous arrêter au milieu d'une phrase. »

2 Étude sur l'interaction dans les listes

2.1 Lors de l'accueil des participants

« *Dans cette expérience, vous allez tester plusieurs interactions avec des listes qui ont été développées au laboratoire. Au cours de cette expérience, des mots vont vous être lus. Vous devez chercher un mot en particulier parmi d'autres et ce le plus rapidement possible. Vous n'êtes pas évalués au cours de ces expériences anonymes. Ce sont les différentes interactions qui seront comparés. On vous demandera votre préférence à la fin de l'expérience.* »

2.2 Lors de la phase de familiarisation

« *Vous allez vous familiariser avec l'interaction [X]. N'hésitez pas à déplacer votre doigt sur l'écran pour voir comment fonctionne cette dernière.* »

2.3 Lors de la phase d'évaluation

« *Nous allons vous demander de chercher un mot le plus rapidement possible, tout en faisant le moins d'erreurs possible, parmi plusieurs.* »

3 Étude sur les mots courts

3.1 Lors de l'accueil des participants

« *Dans cette expérience, vous allez tester plusieurs interactions avec des listes qui ont été développées au laboratoire. Au cours de cette expérience, des mots vont vous être lus. Vous devez chercher un mot en particulier parmi d'autres et ce le plus rapidement possible. Vous n'êtes pas évalués au cours de ces expériences anonymes. Ce sont les différentes interactions qui seront comparées. On vous demandera votre préférence à la fin de l'expérience.* »

3.2 Lors de la phase de familiarisation

3.2.1 Pour la technique à frappe double

« *Choisissez l'initiale du mot que vous souhaitez entrer en maintenant votre doigt sur l'écran. Relâchez, puis appuyez avec deux doigts sur l'écran pour accéder à la liste contenant votre mot.* »

3.2.2 Pour la technique à deux doigts

« *Choisissez l'initiale du mot que vous souhaitez entrer en maintenant votre doigt sur l'écran. Appuyez avec un deuxième doigt sur l'écran, puis relâchez pour accéder à la liste contenant votre mot.* »

3.2.3 Pour la technique glisser-relâcher

« *Choisissez l'initiale du mot que vous souhaitez entrer, puis, tout en maintenant votre doigt sur l'écran, validez avec un deuxième doigt. Toujours sans relâcher votre doigt de l'écran, glissez ensuite sur les autres lettres du mot à saisir, et validez chacune d'entre elles avec un deuxième doigt.* »

4 Étude longue

4.1 Lors de l'accueil des participants

« *Dans cette expérience, vous allez tester deux systèmes de saisie de texte expérimentaux qui ont été développés au laboratoire. Pour cela, on va vous demander d'écrire des phrases qui vous seront dictées. Vous n'êtes pas évalués au cours de ces expériences anonymes. Ce sont les systèmes de saisie qui seront comparés. On vous demandera donc votre préférence à la fin des expériences.* »

4.2 Lors de la phase de familiarisation

« *Vous allez vous familiariser avec les deux systèmes de saisie de texte et les fonctions du système. Pour cela vous aurez plusieurs instructions à suivre et des phrases à recopier. Soyez tranquille, vous pouvez recommencer cette phase plusieurs fois jusqu'à vous sentir à l'aise.* »

4.2.1 Lors de la première utilisation du clavier VODKA

« *Vous allez maintenant vous familiariser avec la saisie d'un mot.*
Pour taper un mot...
Vous allez maintenant vous familiariser avec la saisie d'une phrase
Pour finir une phrase... »

4.2.2 Lors de la première utilisation du clavier DUCK

« *Vous allez maintenant vous familiariser avec la saisie d'un mot.*
Pour taper un mot...
Pour valider un mot, vous devez taper l'écran avec 2 doigts
Vous allez maintenant vous familiariser avec la saisie d'une phrase
Pour finir une phrase... »

4.3 Lors de la phase d'évaluation

« *Nous allons vous demander de recopier le plus rapidement possible, tout en faisant le moins d'erreurs possibles, plusieurs phrases dictées par le système. Afin de maximiser votre vitesse de frappe, nous vous conseillons de mémoriser chaque phrase avant de commencer la saisie. Vous pouvez à tout moment réécouter l'énoncé si vous n'en êtes pas sûr en faisant glisser deux doigts vers le haut ; et écouter ce que vous avez tapé en faisant glisser deux doigts vers le bas. Il est demandé de taper uniquement les caractères alphabétiques, sans se soucier de la ponctuation, de la casse, de l'orthographe, des espaces ou des majuscules.*
Nous vous demandons de maintenir une concentration optimale : l'expérience dure

approximativement vingt minutes. Vous pouvez vous arrêter à tout moment entre deux tâches de recopie, mais nous vous demandons de le faire préférentiellement lorsque vous avez terminé la saisie d'une phrase, avant de commencer la phrase suivante. Veuillez ne pas vous arrêter au milieu d'une phrase. »

Annexe B

Première tentative de réduction du problème d'exploration par l'utilisation d'arbres

Le premier prototype de clavier que nous avions considéré s'est basé sur une étude du clavier H4Writer de Mackenzie[MacKenzie et al., 2011] : en effet, réduire les touches du clavier à un nombre minimal (4 dans le cas de H4Writer) permet de réduire la charge cognitive quant à la frappe de l'utilisateur pour la recherche de caractères à l'écran, et donc d'améliorer sa saisie.

Nous avons donc commencé par concevoir un prototype basé sur une compression de Huffmann, similaire à l'algorithme utilisé par MacKenzie pour réaliser H4Writer.

L'idée est d'optimiser la phase de recherche en construisant des arbres optimisés pour éviter une longue recherche à l'utilisateur, au détriment d'une suite de touches à entrer plus importante.

Les arbres que nous avons réalisés s'appuient sur une compression classique : les lettres les plus utilisées en français se retrouvent directement accessibles. Une des conséquences d'une telle transformation permet de normaliser aux extrêmes : les lettres les plus utilisées deviennent de plus en plus rapides tandis que les lettres les moins utilisées sont plus lentes. C'est une solution qui est très efficace dans les

meilleurs cas, mais qui peut être radicalement handicapante pour un mot complexe.

Nous avons conçu un premier prototype sur un téléphone Samsung Galaxy SII que nous avons testé pour quelques phrases usuelles.

Lors du test, nous nous sommes rendus compte que le clavier en tant que tel n'était pas utilisable :

1. Les arbres utilisés sont trop longs, et donc assez lents à déployer.
2. Il est difficile pour un utilisateur déficient visuel de se représenter facilement l'arbre en question sans entraînement conséquent.
3. L'utilisation d'arbres plus simples implique des arbres moins optimisés : il est alors nécessaire de passer plus de temps pour explorer les touches et la disposition, ce qui rend le principe de H4-Writer inutile.

Le retour que nous avons eu de nos sujets lors de cette étude souligne les mêmes défauts que ceux observés, mais également plusieurs autres problèmes :

1. En cas d'erreur sur une des propositions des feuilles de l'arbre de Huffmann, il devient impossible pour l'utilisateur de corriger facilement. Ce n'est pas problématique pour une lettre facilement accessible comme "e", mais les lettres placées avec plus de difficultés
2. Même si la disposition obtenue était optimale, elle n'est pas facilement adaptable aux besoins de l'utilisateur et est très dépendante du contexte de saisie.

Annexe C

Génération de corpus

Afin de pouvoir tester les différents systèmes au cours de nos études successives, particulièrement dans le cas d'expériences au cours desquelles les utilisateurs devaient taper des phrases au moyen des claviers, nous avons dû nous pencher sur la question du texte source.

Les principaux travaux réalisés en matière de texte source sont des travaux très proches de l'analyse du langage. De plus, la majorité des textes existants et utilisés sont spécifiquement rédigés pour la langue anglaise.

Nous présentons ici la méthodologie que nous avons employée pour réaliser notre corpus de texte. Nos travaux s'appuient intensément sur l'étude de MacKenzie, *Phrase sets for evaluating text entry techniques*[MacKenzie and Soukoreff, 2003].

Bien que l'étude et le corpus qu'il propose concerne uniquement les textes en langue anglaise, les bases et les remarques de Mackenzie sont généralisables au français. En effet, l'utilisation d'un texte clé pour la saisie des phrases implique deux critères de validité dudit texte. D'une part, ce texte doit être valide en *interne*, c'est à dire qu'il ne doit pas y avoir d'effet de l'expérience. Cela implique deux principales caractérisitiques : le texte ne doit pas pouvoir être appris facilement (donc, une grande quantité de contenu), et ne doit pas présenter de répétition (ce qui impose également un contenu varié). Et d'autre part, ce texte doit avoir une validité *externe* en se conformant à un usage typique quotidien. Cela impose l'usage de phrase courtes

et sensées.

Idéalement, le corpus doit être validé par une étude empirique, comme mentionné par Mackenzie[MacKenzie and Soukoreff, 2003] et Kano[Kano et al., 2006]. Afin de maximiser le temps des sujets et de généraliser notre étude, nous avons choisi d'opter pour une validation théorique.

Pour cela, nous avons utilisé l'indice de lisibilité de Flesch-Kincaid pour mesurer le niveau de langue des textes. Cet indice permet de valider la lisibilité d'un texte et d'indiquer sa difficulté de lecture. Dans sa version française, l'indice est décliné sous le nom d'indice de Flesch-Mesnager[MESNAGER, 1989]. Bien que fortement décrié au profit d'une analyse textuelle, l'indice de Flesch permet cependant d'obtenir une valeur de la "lisibilité" indépendante de tout texte de référence .

$$I = 212 - (0,7W + S)$$

Couplé à cet outil, nous avons également repris la démarche de MacKenzie pour son approche fréquentielle au niveau des lettres et des bigrammes sur la langue. Nous avons réalisé un programme qui permet de calculer la concordance d'un corpus à nos besoins en tenant compte des contraintes d'expérimentation suivantes :

— Orthographe simple (sujets) ;
— Minimiser le nombre de mots à connaître ;
— Avoir suffisament de contenu ;
— Proche d'une saisie quotidienne ;
— Aucun mot avec ligatures, accents ou autre ;
— Respect de la répartition des mots.

1 Outil : `corpus_study.py`

— Répartition fréquentielle
— Corrélation bigrammes et lettres
— n-mots les plus utilisés
— Indice de Flesch-Mesnager

2 Corpus proposé

Mots les plus utilisés : le, la, les, il, de, est, ne, pas, un, a, une, sont, des, dans, que, se, on, plus, elle, toujours

Longueur de mot minimale (LMM) : 1

Longueur de mot maximale (LMX) : 13

Longueur de mot moyenne (LMY) : 4,21

Nombre de mots (NM) : 727

Corrélation fréquence bigramme (CFB) : 80,12%

Corrélation fréquence lettre (CFL) : 98,26%

Indice de Flesch-Ménager (IFM) : 100

Table C.1: Corpus utilisé dans nos expériences

Phrase	**LMM**	**LMX**	**LMY**	**NM**	**CFB**	**CFL**	**IFM**
Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
elles lisent des belles histoires	3	9	5,80	5	59,74%	76,91%	67,00
il faut garder une poire pour la soif	2	6	3,75	8	21,93%	77,06%	100,00
a quelque chose malheur est bon	1	7	4,33	6	24,21%	79,03%	100,00
le docteur exige que vous vous reposiez	2	8	4,71	7	21,44%	76,84%	85,00
elle a pris ses jouets et son ours	1	6	3,38	8	39,69%	82,11%	100,00
il se tenait immobile dans la cour	2	8	4,00	7	26,20%	85,24%	95,00
un malheur ne vient jamais seul	2	7	4,33	6	26,08%	84,61%	100,00
vous sortez plus souvent le dimanche	2	8	5,17	6	23,84%	84,30%	89,33
on parle mal quand on ne veut rien dire	2	5	3,44	9	34,03%	83,09%	100,00
les griffes du chat grattent le tapis	2	8	4,43	7	39,12%	80,42%	100,00
il a pris les affaires de sa copine	1	8	3,50	8	48,27%	78,90%	99,00
la belle plume fait le bel oiseau	2	6	3,86	7	29,35%	72,57%	100,00
elle ne peut pas deviner mon origine	2	7	4,29	7	25,70%	85,97%	75,00
il doit y avoir une explication logique	1	11	4,71	7	8,27%	68,55%	65,00
la terre tremblait quand elle criait	2	9	5,17	6	26,24%	80,70%	100,00
il ne veut pas partir avec moi	2	6	3,43	7	1,40%	83,86%	100,00
le clown a fait rire les enfants	1	7	3,71	7	56,15%	83,80%	100,00
notre tombeau est le cœur des hommes	2	7	4,29	7	41,08%	82,45%	100,00

Suite sur la page suivante

Table C.1 – suite de la page précédente

Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
vous pouvez venir chercher vos clefs	3	8	5,17	6	20,89%	64,69%	100,00
il a le nez rouge comme un clown	1	5	3,12	8	17,61%	69,03%	100,00
tour le monde a son grain de folie	1	5	3,38	8	35,29%	82,49%	100,00
tu veux bien acheter un kilo de sucre	2	7	3,75	8	37,13%	83,29%	99,00
vous aimez entendre le murmure du vent	2	8	4,57	7	48,80%	81,55%	85,00
les filles de cette villes sont grandes	2	7	4,71	7	56,97%	82,52%	95,00
elle prend une grande enveloppe	3	9	5,40	5	33,58%	72,86%	53,00
le cowboy joue du banjo au saloon	2	6	3,86	7	14,20%	48,40%	100,00
il vaut mieux tenir que courir	2	6	4,17	6	19,22%	71,50%	100,00
la voiture est sortie de la route	2	7	3,86	7	32,31%	87,99%	95,00
la pomme ne tombe pas loin de la branche	2	7	3,56	9	1,60%	78,30%	100,00
je prends un panier pour les courses	2	7	4,29	7	44,49%	80,60%	100,00
chacun de mes actes est une destruction	2	11	4,71	7	42,17%	83,65%	85,00
la petite fille joue dans la cour	2	6	3,86	7	19,23%	85,62%	100,00
les mots ne se battent pas sur le papier	2	7	3,56	9	41,06%	87,92%	100,00
ils dessinent des oiseaux rouges	3	9	5,60	5	53,61%	81,24%	81,00
il faut laver son linge sale en famille	2	7	4,00	8	34,32%	78,20%	90,25
les enfants et les fous sont devins	2	7	4,14	7	61,58%	77,76%	100,00
le bonheur est le plaisir des sages	2	7	4,14	7	51,86%	86,62%	100,00
une grande douleur nous rend plus forts	3	7	4,71	7	25,87%	74,44%	100,00

Suite sur la page suivante

Table C.1 – suite de la page précédente

Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
le pouvoir sans abus perd son charme	2	7	4,29	7	14,82%	80,47%	100,00
la raison est ce qui effraie chez un fou	2	7	3,56	9	20,52%	82,80%	100,00
il aime aller courir dans les bois	2	6	4,00	7	32,63%	79,21%	100,00
tu peux tout faire seul si tu insistes	2	8	3,88	8	26,07%	76,09%	100,00
ma tante veut que je parte avec elle	2	5	3,62	8	25,45%	78,74%	99,00
il pense que tu as toujours raison	2	8	4,00	7	24,58%	82,24%	100,00
une hirondelle ne fait pas le printemps	2	10	4,71	7	40,07%	88,76%	85,00
il est absurde de ne pas vouloir changer	2	7	4,12	8	23,82%	93,35%	99,00
une baleine a fait un saut spectaculaire	1	13	4,86	7	19,19%	84,71%	65,00
le monde est le plus beau spectacle	2	9	4,14	7	39,93%	83,07%	100,00
les chiens ne font pas des chats	2	6	3,71	7	47,10%	76,11%	100,00
la voile se gonfle dans le vent	2	6	3,57	7	40,61%	77,33%	100,00
ta nouvelle voisine est vraiment belle	2	8	5,50	6	40,70%	86,52%	66,00
fou qui a le choix et prend le pire	1	5	3,00	9	40,14%	80,78%	100,00
les voyages forment la jeunesse	2	8	5,40	5	57,64%	82,78%	81,00
tout usage finit par se changer en abus	2	7	4,00	8	13,47%	84,90%	90,25
la poule a pondu un œuf au poulailler	1	10	3,75	8	28,73%	55,68%	99,00
les loups ne se mangent pas entre eux	2	7	3,75	8	52,90%	85,67%	100,00
il ne pourra pas remporter la victoire	2	9	4,57	7	18,89%	78,08%	85,00
il aime naviguer sur son voilier	2	8	4,50	6	12,96%	76,33%	89,33

Suite sur la page suivante

Table C.1 – suite de la page précédente

Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
il avait toujours un moment pour dormir	2	8	4,71	7	22,79%	63,29%	95,00
que de choses il faut ignorer pour agir	2	7	4,00	8	37,76%	81,90%	99,00
la nuit tous les chats sont gris	2	5	3,71	7	36,27%	69,76%	100,00
on a toujours plus de bien que de vie	1	8	3,22	9	31,31%	82,61%	100,00
tous les fous sont dans les asiles	3	6	4,00	7	45,60%	69,29%	100,00
il faut que la jeunesse se passe	2	8	3,71	7	22,36%	80,85%	100,00
je vais pouvoir refaire toute la maison	2	7	4,71	7	25,27%	83,55%	85,00
plus on est de fous plus on rit	2	4	3,00	8	25,58%	70,83%	100,00
le hasard gouverne nos actions	2	8	5,20	5	20,06%	83,76%	81,00
le chemin du devoir est toujours proche	2	8	4,71	7	22,29%	79,96%	95,00
tu peux passer des vacances tranquilles	2	11	5,67	6	40,07%	86,85%	77,67
la souris danse quand le chat est loin	2	6	3,88	8	20,38%	85,40%	100,00
le soleil est le fourneau des pauvres	2	8	4,43	7	49,40%	87,96%	100,00
mesure la profondeur avant de plonger	2	10	5,33	6	41,15%	82,71%	66,00
je trouve un pot de confiture de fraise	2	9	4,00	8	37,06%	83,92%	90,25
chien qui aboie ne mord jamais	2	6	4,17	6	15,88%	72,51%	100,00
la vache regarde passer les trains	2	7	4,83	6	41,91%	81,12%	89,33
il y a un bruit infernal dans la rue	1	8	3,11	9	-4,43%	71,29%	100,00
qui aime la mer aime aussi les bateaux	2	7	3,88	8	26,33%	77,40%	99,00
elle pointe un doigt dans sa direction	2	9	4,57	7	26,27%	84,60%	95,00

Suite sur la page suivante

Table C.1 – suite de la page précédente

Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
il est toujours content de ses cadeaux	2	8	4,57	7	56,19%	86,43%	100,00
les vices empoisonnent les plaisirs	3	12	6,20	5	52,20%	80,62%	67,00
la porte donne directement sur la cour	2	11	4,57	7	30,99%	88,78%	85,00
je suis le moins rapide sur ce circuit	2	7	3,88	8	11,23%	79,89%	100,00
les phares guident les navires la nuit	2	7	4,57	7	60,77%	91,18%	95,00
chassez le naturel il revient au galop	2	7	4,57	7	32,61%	89,26%	85,00
la terre tourne autour du soleil	2	6	4,50	6	23,30%	81,73%	89,33
elles croient toujours aux miracles	3	8	6,20	5	50,09%	87,33%	81,00
est assez riche qui ne doit rien	2	5	3,71	7	27,06%	85,14%	100,00
les louanges sont le protocole des sots	2	9	4,71	7	53,94%	74,24%	95,00
ils sortent plus souvent le dimanche	2	8	5,17	6	35,56%	89,39%	89,33
le chien me regarde en remuant la queue	2	7	4,00	8	44,47%	82,43%	100,00
il ne faut pas hurler avec les loups	2	6	3,62	8	38,10%	83,37%	100,00
la jeunesse est une ivresse continuelle	2	11	5,67	6	51,60%	87,28%	42,67
les conseillers ne sont pas les payeurs	2	11	4,71	7	47,44%	81,65%	100,00
les conseilleurs ne sont pas les payeurs	2	12	4,86	7	41,96%	82,79%	100,00
il est content de cette trouvaille	2	10	4,83	6	54,15%	81,69%	89,33
on se lasse de tout sauf de comprendre	2	10	3,88	8	37,11%	87,95%	100,00
il ne vient au bureau que le mercredi	2	8	3,75	8	41,95%	83,43%	100,00
sans amour les fautes sont grandes	3	7	4,83	6	47,23%	77,54%	100,00

Suite sur la page suivante

Table C.1 – suite de la page précédente

Phrase	LMM	LMX	LMY	NM	CBF	CLF	IFM
les bons comptes font les bons amis	3	7	4,14	7	48,95%	66,47%	100,00
on ne peut pas discuter dans le noir	2	8	3,62	8	17,99%	90,24%	100,00
le soleil luit pour tout le monde	2	6	3,86	7	32,59%	73,64%	100,00
la roulette russe est un jeu dangereux	2	9	4,57	7	33,57%	85,08%	85,00
mon voisin de chambre est gentil	2	7	4,50	6	39,98%	83,39%	100,00

Annexe D

Liste des mots courts

Nous présentons ici la liste des mots courts que nous avons réalisée suite à l'analyse des proportions d'usage de chacun au sein de la langue française.

Table D.1 – Liste de mots courts utilisée par les différents prototypes

a	au	avec	aux	ai	as	avez		
b	bien	bon						
c	ce	ces	cela	chez	cet	car	ceux	
d	de	des	dans	du	dit	dont	déjà	donc
e	et	en	est	eux	eu	es		
f	fait	fois	faut	fais	fini			
g	gens	gros	goût	gris	gare	gars		
h	haut	huit	hors	hier				
i	il	ils	ici	idée	ira	irai		
j	je	jour	joie	joue	jeu			
k	képi	kilo						
l	la	le	les	lui	leur			
m	mais	même	me	mon	moi	ma	mes	
n	ne	nous	non	ni	nos	nuit		
o	on	ou	ont	oui	or			
p	pas	pour	par	plus	peut	peu	puis	
q	que	qui	qu'	quoi	quel			
r	rien	rue	rôle	rire				
s	se	sur	son	sa	ses	si		
t	tout	tu	très	tous	trop	te	toi	ton
u	un	une						
v	vous	vers	voir	va	vu	veux	voit	vais

Bibliographie

[Accot and Zhai, 2002] Accot, J. and Zhai, S. (2002). More than dotting the i's—foundations for crossing-based interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80. ACM.

[Al Faraj et al., 2009] Al Faraj, K., Mojahid, M., and Vigouroux, N. (2009). *BigKey : A Virtual Keyboard for Mobile Devices*, pages 3–10. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Amblard and Boumaza, 2014] Amblard, M. and Boumaza, A. (2014). Robots humains, avez-donc une réalité ? *artificial intelligence : Philosophy, Geoinformatics and Law*, page 14.

[Amma et al., 2016] Amma, C., Georgi, M., Lenz, T., and Winnen, F. (2016). Kinemic wave : A mobile freehand gesture and text-entry system. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 3639–3642, New York, NY, USA. ACM.

[Arif and Stuerzlinger, 2009] Arif, A. S. and Stuerzlinger, W. (2009). Analysis of text entry performance metrics. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 100–105. IEEE.

[Aulagner et al., 2010] Aulagner, G., François, R., Martin, B., Michel, D., and Raynal, M. (2010). Floodkey : Increasing software keyboard keys by reducing needless ones without occultation. In *Proceedings of the 10th WSEAS International Conference on Applied Computer Science*, ACS'10, pages 412–417, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

[Ausbrooks et al., 2003] Ausbrooks, R., Buswell, S., Carlisle, D., Dalmas, S., Devitt, S., Diaz, A., Froumentin, M., Hunter, R., Ion, P., Kohlhase, M., et al. (2003). Mathematical markup language (mathml) version 2.0 . w3c recommendation. *World Wide Web Consortium*, 2003.

[Azenkot and Lee, 2013] Azenkot, S. and Lee, N. B. (2013). Exploring the use of speech input by blind people on mobile devices. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 11 :1–11 :8, New York, NY, USA. ACM.

[Azenkot et al., 2012] Azenkot, S., Wobbrock, J. O., Prasain, S., and Ladner, R. E. (2012). Input finger detection for nonvisual touch screen text entry in perkinput. In *Proceedings of Graphics Interface (GI '12)*, number d, pages 121–129.

[Bi et al., 2013] Bi, X., Azenkot, S., Partridge, K., and Zhai, S. (2013). Octopus : Evaluating touchscreen keyboard correction and recognition algorithms via. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 543–552, New York, NY, USA. ACM.

[Bi et al., 2012] Bi, X., Chelba, C., Ouyang, T., Partridge, K., and Zhai, S. (2012). Bimanual gesture keyboard. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 137–146, New York, NY, USA. ACM.

[Bonner et al., 2010] Bonner, M. N., Brudvik, J. T., Abowd, G. D., and Edwards, W. K. (2010). No-look notes : Accessible eyes-free multi-touch text entry. In *Proceedings of the 8th International Conference on Pervasive Computing*, Pervasive'10, pages 409–426, Berlin, Heidelberg. Springer-Verlag.

[Brooke et al., 1996] Brooke, J. et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194) :4–7.

[Cao et al., 2009] Cao, A., Chintamani, K. K., Pandya, A. K., and Ellis, R. D. (2009). Nasa tlx : Software for assessing subjective mental workload. *Behavior research methods*, 41(1) :113–117.

[Castellucci and MacKenzie, 2008] Castellucci, S. J. and MacKenzie, I. S. (2008). Graffiti vs. unistrokes : an empirical comparison. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 305–308. ACM.

[Chaudhuri,] Chaudhuri, S. raille comes unbound from the book : how technology can stop a literary crisis. `https://www.theguardian.com/society/2012/feb/14/technology-brings-braille-back-apple`. Consulté : 2016-08-04.

[Chen et al., 2014] Chen, X. A., Grossman, T., and Fitzmaurice, G. (2014). Swipeboard : A text entry technique for ultra-small interfaces that supports novice to expert transitions. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 615–620, New York, NY, USA. ACM.

[Cooper, 2012] Cooper, W. E. (2012). *Cognitive aspects of skilled typewriting*. Springer Science & Business Media.

[Copestake, 1997] Copestake, A. (1997). Augmented and alternative nlp techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*, pages 37–42.

[Crossman, 1959] Crossman, E. (1959). A theory of the acquisition of speed-skill. *Ergonomics*, 2(2) :153–166.

[De Marneffe et al., 2006] De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

[Even-Zohar and Roth, 2000] Even-Zohar, Y. and Roth, D. (2000). A classification approach to word prediction. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 124–131. Association for Computational Linguistics.

[Garay-Vitoria and Abascal, 2006] Garay-Vitoria, N. and Abascal, J. (2006). Text prediction systems : a survey. *Universal Access in the Information Society*, 4(3) :188–203.

[Gentner et al., 1983] Gentner, D. R., Grudin, J. T., Larochelle, S., Norman, D. A., and Rumelhart, D. E. (1983). A glossary of terms including a classification of typing errors. In *Cognitive aspects of skilled typewriting*, pages 39–43. Springer.

[Greenberg et al., 1995] Greenberg, S., Darragh, J. J., Maulsby, D., and Witten, I. H. (1995). Extra-ordinary human-computer interaction. chapter Predictive Interfaces : What Will They Think of Next ?, pages 103–140. Cambridge University Press, New York, NY, USA.

[Hart and Staveland, 1988] Hart, S. G. and Staveland, L. E. (1988). Development of nasa-tlx (task load index) : Results of empirical and theoretical research. *Advances in psychology*, 52 :139–183.

[Hassenzahl et al., 2003] Hassenzahl, M., Burmester, M., and Koller, F. (2003). Attrakdiff : Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In *Mensch & Computer 2003*, pages 187–196. Springer.

[Hatzigiannakoglou and Kampouraki, 2016] Hatzigiannakoglou, P. D. and Kampouraki, M. T. (2016). An accessible keyboard for android devices as a means for promoting braille literacy. *International Journal of Interactive Mobile Technologies (iJIM)*, 10(2) :77–78.

[Helle et al., 2001] Helle, S., LePlâtre, G., Marila, J., and Laine, P. (2001). Menu sonification in a mobile phone – a prototype study. pages 255–260.

[Hempstalk, 2006] Hempstalk, K. (2006). The great keyboard debate : Qwerty versus dvorak. In *Proceedings of NZCSRSC'07, the Fifth New Zealand Computer Science Research Student Conference*.

[Heron et al., 2014] Heron, J., Bosse, J., He, Q., Gao, Y., Trassin, M., Ye, L., Clarkson, J., Wang, C., Liu, J., Salahuddin, S., et al. (2014). Deterministic switching of ferromagnetism at room temperature using an electric field. *Nature*, 516(7531) :370–373.

[Hick, 1952] Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1) :11–26.

[Hoggan et al., 2008] Hoggan, E., Brewster, S. A., and Johnston, J. (2008). Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proceedings*

of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, pages 1573–1582, New York, NY, USA. ACM.

[Hong et al., 2015] Hong, J., Heo, S., Isokoski, P., and Lee, G. (2015). Splitboard : A simple split soft keyboard for wristwatch-sized touch screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1233–1236, New York, NY, USA. ACM.

[Human Performance Group NASA Ames Research Center, 1986] Human Performance Group NASA Ames Research Center, N. (1986). *NASA Task Load Index (NASA-TLX)*. NASA.

[Hyman, 1953] Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Journal of experimental psychology*, 45(3) :188.

[International Organization for Standardization, 1998] International Organization for Standardization, I. (1998). *ISO 9241-11 : Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) : Part 11 : Guidance on Usability*.

[Kamel and Landay, 2000] Kamel, H. M. and Landay, J. A. (2000). A study of blind drawing practice : Creating graphical information without the visual channel. In *Proceedings of the Fourth International ACM Conference on Assistive Technologies*, Assets '00, pages 34–41, New York, NY, USA. ACM.

[Kane et al., 2008] Kane, S. K., Bigham, J. P., and Wobbrock, J. O. (2008). Slide rule : making mobile touch screens accessible to blind people using multi-touch interaction techniques. *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility - Assets '08*, pages 73–80.

[Kano et al., 2006] Kano, A., Read, J. C., and Dix, A. (2006). Children's phrase set for text input method evaluations. In *Proceedings of the 4th Nordic conference on Human-computer interaction : changing roles*, pages 449–452. ACM.

[Kinkead, 1975] Kinkead, R. (1975). Typing speed, keying rates, and optimal keyboard layouts. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 19, pages 159–161. SAGE Publications.

[Kjeldskov and Stage, 2004] Kjeldskov, J. and Stage, J. (2004). New techniques for usability evaluation of mobile systems. *International journal of human-computer studies*, 60(5) :599–620.

[Kristensson and Zhai, 2004] Kristensson, P.-O. and Zhai, S. (2004). Shark2 : A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 43–52, New York, NY, USA. ACM.

[Leporini et al., 2012] Leporini, B., Buzzi, M. C., and Buzzi, M. (2012). Interacting with mobile devices via voiceover : Usability and accessibility issues. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 339–348, New York, NY, USA. ACM.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[Liebowitz and Margolis, 1990] Liebowitz, S. J. and Margolis, S. E. (1990). The fable of the keys. *Journal of Law and Economics*, pages 1–25.

[Lindquist et al., 2007] Lindquist, D., Denning, T., Kelly, M., Malani, R., Griswold, W. G., and Simon, B. (2007). Exploring the potential of mobile phones for active learning in the classroom. *SIGCSE Bull.*, 39(1) :384–388.

[Lucero et al., 2010] Lucero, A., Keränen, J., and Korhonen, H. (2010). Collaborative use of mobile phones for brainstorming. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, pages 337–340, New York, NY, USA. ACM.

[MacKenzie, 2002a] MacKenzie, I. S. (2002a). Kspc (keystrokes per character) as a characteristic of text entry techniques. In *International Conference on Mobile Human-Computer Interaction*, pages 195–210. Springer.

[MacKenzie and Soukoreff, 2003] MacKenzie, I. S. and Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 754–755. ACM.

[MacKenzie et al., 2011] MacKenzie, I. S., Soukoreff, R. W., and Helga, J. (2011). 1 thumb, 4 buttons, 20 words per minute : Design and evaluation of h4-writer. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 471–480. ACM.

[MacKenzie and Zhang, 1997] MacKenzie, I. S. and Zhang, S. X. (1997). The immediate usability of graffiti. In *Graphics Interface*, volume 97, pages 129–137.

[MacKenzie and Zhang, 1999a] MacKenzie, I. S. and Zhang, S. X. (1999a). The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 25–31. ACM.

[MacKenzie and Zhang, 1999b] MacKenzie, I. S. and Zhang, S. X. (1999b). The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, number May, pages 25–31, New York, New York, USA. ACM Press.

[MacKenzie, 2002b] MacKenzie, S. (2002b). Mobile text entry using three keys. In *Proceedings of the second Nordic conference on Human-computer interaction*, pages 27–34. ACM.

[Madison, 2012] Madison, J. (2012). *Damn you, autocorrect!* Random House.

[Mankoff and Abowd, 1998] Mankoff, J. and Abowd, G. D. (1998). Cirrin : A word-level unistroke keyboard for pen input. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, pages 213–214, New York, NY, USA. ACM.

[Marek et al., 2016] Marek, T., Krejcar, O., and Selamat, A. (2016). Possibilities for development and use of 3d applications on the android platform. In *Asian Conference on Intelligent Information and Database Systems*, pages 519–529. Springer.

[Mattheiss et al., 2015] Mattheiss, E., Regal, G., Schrammel, J., Garschall, M., and Tscheligi, M. (2015). Edgebraille : Braille-based text input for touch devices. *Journal of Assistive Technologies*, 9(3) :147–158.

[Merlin and Raynal, 2009] Merlin, B. and Raynal, M. (2009). Spreadkey : increasing software keyboard key by recycling needless ones. In *Proc. of AAATE*, pages 138–143.

[Merlin and Raynal, 2012] Merlin, B. and Raynal, M. (2012). Soft keyboard evaluations : Integrating user's background. *Cognitively Informed Intelligent Interfaces : Systems Design and Development : Systems Design and Development*, page 21.

[MESNAGER, 1989] MESNAGER, J. (1989). Lisibilité des textes pour enfants : un nouvel outil ? *Communication et Langages*, (79) :18–38.

[Nicolau et al., 2015] Nicolau, H., Montague, K., Rodrigues, A., Guerreiro, T., and Hanson, V. L. (2015). Typing performance of blind users : An analysis of touch behaviors, learning effect, and in-situ usage. *Proceedings of ASSETS'15*.

[Norberg, 1990] Norberg, A. L. (1990). High-technology calculation in the early 20th century : Punched card machinery in business and government. *Technology and Culture*, 31(4) :753–779.

[Oliveira et al., 2011] Oliveira, J. a., Guerreiro, T., Nicolau, H., Jorge, J., and Gonçalves, D. (2011). Blind people and mobile touch-based text-entry. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility - ASSETS '11*, page 179, New York, New York, USA. ACM Press.

[Oney et al., 2013] Oney, S., Harrison, C., Ogan, A., and Wiese, J. (2013). Zoomboard : A diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2799–2802, New York, NY, USA. ACM.

[Partridge et al., 2002] Partridge, K., Chatterjee, S., Sazawal, V., Borriello, G., and Want, R. (2002). Tilttype : accelerometer-supported text entry for very small devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 201–204. ACM.

[Perlin, 1998] Perlin, K. (1998). Quikwriting : continuous stylus-based text entry. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 215–216. ACM.

[Raynal, 2006] Raynal, M. (2006). Claviers gag : claviers logiciels optimisés pour la saisie de texte au stylet. In *Proceedings of the 18th Conference on l'Interaction Homme-Machine*, pages 3–10. ACM.

[Raynal, 2007] Raynal, M. (2007). Le système keyglass : Système d'ajout dynamique de touches sur clavier logiciel. *Traitement Automatique des Langues, Communication assistée*, 48(2) :97–121.

[Raynal et al., 2014] Raynal, M., MacKenzie, I. S., and Merlin, B. (2014). Semantic keyboard : Fast movements between keys of a soft keyboard. In *International Conference on Computers for Handicapped Persons*, pages 195–202. Springer.

[Raynal and Truillet, 2007] Raynal, M. and Truillet, P. (2007). Fisheye keyboard : whole keyboard displayed on pda. In *International Conference on Human-Computer Interaction*, pages 452–459. Springer.

[Raynal and Vigouroux, 2005a] Raynal, M. and Vigouroux, N. (2005a). Genetic algorithm to generate optimized soft keyboard. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1729–1732. ACM.

[Raynal and Vigouroux, 2005b] Raynal, M. and Vigouroux, N. (2005b). Keyglasses : Semi-transparent keys to optimize text input on virtual keyboard. In *Assistive technology : from virtuality to reality-8th European conference for the advancement of assistive technology in europe (AAATE 2005), Lille, France*, volume 6, pages 05–09.

[Ren et al., 2008] Ren, K., Liu, S., Lin, M., Wang, Y., and Zhang, Q. (2008). A compact electroactive polymer actuator suitable for refreshable braille display. *Sensors and Actuators A : Physical*, 143(2) :335 – 342.

[Ristad and Yianilos, 1998] Ristad, E. S. and Yianilos, P. N. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5) :522–532.

[Sánchez and Aguayo, 2007] Sánchez, J. and Aguayo, F. (2007). Mobile messenger for the blind. In *Universal Access in Ambient Intelligence Environments*, pages 369–385. Springer.

[Šepić et al., 2014] Šepić, B., Ghanem, A., and Vogel, S. (2014). Brailleeasy : One-handed braille keyboard for smartphones. *Studies in health technology and informatics*, 217 :1030–1035.

[Shabnam and Govindarajan, 2014] Shabnam, M. and Govindarajan, S. (2014). Braille-coded gesture patterns for touch-screens a character input method for differently enabled persons using mobile devices. In *Proceedings on International Conference on Communication, Computing and Information Technology ICCC-MIT*, volume 1, pages 1–5. Citeseer.

[Shahzadi et al., 2011] Shahzadi, S., Fatima, B., and Kamran, M. (2011). Urdu t9 and word prediction messaging system for android : Ut9wp.

[Shibata et al., 2016] Shibata, T., Afergan, D., Kong, D., Yuksel, B. F., MacKenzie, I. S., and Jacob, R. J. (2016). Driftboard : A panning-based text entry technique for ultra-small touchscreens. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 575–582, New York, NY, USA. ACM.

[Shneiderman, 2000] Shneiderman, B. (2000). The limits of speech recognition. *Communications of the ACM*, 43(9) :63–65.

[Soukoreff and MacKenzie, 2003] Soukoreff, R. W. and MacKenzie, I. S. (2003). Metrics for text entry research : An evaluation of msd and kspc, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 113–120, New York, NY, USA. ACM.

[Southern et al., 2012] Southern, C., Clawson, J., Frey, B., Abowd, G., and Romero, M. (2012). An evaluation of brailletouch : Mobile touchscreen text entry for the visually impaired. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 317–326, New York, NY, USA. ACM.

[Subash et al., 2012] Subash, N. S., Nambiar, S., and Kumar, V. (2012). Braillekey : An alternative braille text input system : Comparative study of an innovative simplified text input system for the visually impaired. In *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, pages 1–4. IEEE.

[Tinwala et al., 2009] Tinwala, H. et al. (2009). Eyes-free text entry on a touchscreen phone. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 83–88. IEEE.

[Villemonteix and Khaneboubi, 2012] Villemonteix, F. and Khaneboubi, M. (2012). Utilisations de tablettes tactiles à l'école primaire. In *Journées Communication et Apprentissage Instrumentés en Réseau*. Université Picardie Jules Verne.

[Vogelsberger et al., 2014] Vogelsberger, M., Genel, S., Springel, V., Torrey, P., Sijacki, D., Xu, D., Snyder, G. F., Bird, S., Nelson, D., and Hernquist, L. (2014). Properties of galaxies reproduced by a hydrodynamic simulation. *arXiv preprint arXiv :1405.1418*.

[Walker and Kogan, 2009] Walker, B. N. and Kogan, A. (2009). Spearcon performance and preference for auditory menus on a mobile phone. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5615 LNCS(PART 2) :445–454.

[Walmsley et al., 2014] Walmsley, W. S., Snelgrove, W. X., and Truong, K. N. (2014). Disambiguation of imprecise input with one-dimensional rotational text entry. *ACM Transactions on Computer-Human Interaction*, 21(1) :1–40.

[Wobbrock and Myers, 2006] Wobbrock, J. and Myers, B. (2006). Trackball text entry for people with motor impairments. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 479–488. ACM.

[Wobbrock et al., 2003] Wobbrock, J. O., Myers, B. A., and Kembel, J. A. (2003). Edgewrite : A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 61–70, New York, NY, USA. ACM.

[Young, 2014] Young, J. (2014). Akamai releases second quarter 2014 'state of the internet' report. `https://www.akamai.com/fr/fr/about/news/press/2014-press/akamai-releases-second-quarter-2014-state-of-the-internet-report.jsp`. Accessed : 2016-07-14.

[Zhai et al., 2000] Zhai, S., Hunter, M., and Smith, B. A. (2000). The metropolis keyboard-an exploration of quantitative techniques for virtual keyboard design. In

Proceedings of the 13th annual ACM symposium on User interface software and technology, pages 119–128. ACM.

[Zhai et al., 2002] Zhai, S., Hunter, M., and Smith, B. A. (2002). Performance optimization of virtual keyboards. *Human–Computer Interaction*, 17(2-3) :229–269.

[Zhai and Kristensson, 2007] Zhai, S. and Kristensson, P. (2007). Introduction to shape writing.

[Zhai and Kristensson, 2003] Zhai, S. and Kristensson, P.-O. (2003). Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 97–104, New York, NY, USA. ACM.

[Zhai and Kristensson, 2010] Zhai, S. and Kristensson, P. O. (2010). Introduction to shape writing 7. *Text Entry Systems : Mobility, Accessibility, Universality.*

[Zhai et al., 2009] Zhai, S., Kristensson, P. O., Gong, P., Greiner, M., Peng, S. A., Liu, L. M., and Dunnigan, A. (2009). Shapewriter on the iphone : From the laboratory to the real world. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2667–2670, New York, NY, USA. ACM.

[Zhai et al., 2005] Zhai, S., Kristensson, P.-O., and Smith, B. A. (2005). In search of effective text input interfaces for off the desktop computing. *Interacting with computers*, 17(3) :229–250.

[Zhao et al., 2007] Zhao, S., Dragicevic, P., Chignell, M., Balakrishnan, R., and Baudisch, P. (2007). Earpod : Eyes-free menu selection using touch input and reactive audio feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1395–1404, New York, NY, USA. ACM.

Étudiez comme si vous deviez vivre toujours ; vivez comme si vous deviez mourir demain.

Isidore de Séville