Systèmes de saisie de textes pour les personnes handicapées moteur : optimisation, interaction et mesure de l'utilisabilité

Mathieu Raynal

TOULOUSE III

Université Toulouse III **École Doctorale Informatique et Télécommunication**

Formation Doctorale Informatique de l'Image et du Langage

Systèmes de saisie de textes pour les personnes handicapées moteur : *optimisation, interaction et mesure de l'utilisabilité*

THÈSE

présentée et soutenue publiquement le 5 décembre 2005

pour l'obtention du

Doctorat de l'Université Toulouse III – Paul Sabatier

(spécialité informatique)

par

Mathieu Raynal

Composition du jury

Rapporteurs :	Pr Franck Poirier	Universite de Bretagne-Sud
	Pr Jean-Marc Toulotte	Université des Sciences et Technologies de Lille
Examinateurs :	Pr Philippe Palanque	Université Toulouse III
	Mr Stéphane Chatty	IntuiLab
Directeurs de thèse :	Pr Régine André-Obrecht	Université Toulouse III
	Mme Nadine Vigouroux	Institut de Recherche en Informatique de Toulouse

Institut de Recherche en Informatique de Toulouse — UMR 5505

Table des matières

Table des figures **xi**

Liste des tableaux **xv**

Introduction **1**

I Méthodologies d'évaluation des systèmes de saisie de texte 7

Introduction **9**

1 Les lois prédictives **11**

1.1 Temps de recherche visuelle d'une cible 11

1.2 Temps de pointage d'une cible 12

1.2.1 La loi de Fitts 12

1.2.2 Ajustement de la loi de Fitts pour une utilisation en Interaction Homme-Machine 13

1.3 Prédiction des performances d'un clavier 14

1.3.1 Le modèle Keystroke-Level Model 15

1.3.2 Le modèle de Soukoreff et Mackenzie 15

1.3.3 Quelles valeurs pour les constantes pour une évaluation théorique de clavier ? 18

1.3.4 Bilan 21

1.4 Entre loi prédictive et expérimentation : quelle solution ? 22
1.5 Nos choix pour la suite 23

2 Méthodologies d'évaluation **25**
2.1 Les tâches à réaliser 25
2.1.1 La tâche de recopie 26
2.1.2 La tâche de création 28
2.2 Les variables à analyser 28
2.2.1 La vitesse de saisie de texte 29
2.2.2 Les différentes méthodes de calcul du taux d'erreur 29
2.3 Bilan 33
2.4 Nos choix pour la suite 33

3 La plate-forme E-Assiste **37**
3.1 Problématiques 37
3.1.1 Des résultats difficiles à comparer 37
3.1.2 Loin de représenter une utilisation normale 39
3.1.3 Des évaluations pas assez poussées 39
3.2 La plate-forme E-Assiste 40
3.2.1 Objectifs 40
3.2.2 Principe de modularité 40
3.2.3 Architecture 41

Conclusion **45**

II Optimisation de la disposition des caractères 47

Introduction **49**

4 Les méthodes existantes **51**
4.1 Les claviers réduits . . . 51
4.1.1 Les solutions matérielles . . . 52
4.1.2 Solutions logicielles . . . 56
4.2 Analyse de claviers virtuels type téléphone pour un usage par des handicapés moteur . . . 57
4.3 Méthodes d'agencement des caractères . . . 60
4.3.1 Les claviers alphabétiques . . . 60
4.3.2 Agencement réalisé intuitivement . . . 61
4.3.3 Agencement obtenu avec techniques d'optimisation automatiques 62
4.4 Bilan . . . 66

5 Méthode d'optimisation de la disposition des touches et caractères 69

5.1 Optimisation de la disposition des touches . . . 69
5.1.1 Evaluation d'une disposition de touches . . . 70
5.1.2 Les dispositions de touches testées . . . 70
5.1.3 Résultats obtenus . . . 71
5.2 L'agencement des caractères : un problème NP-complet . . . 73
5.2.1 Appartenance à la classe NP . . . 73
5.2.2 Problème NP-difficile . . . 74
5.3 Méthode d'optimisation par algorithme génétique . . . 75
5.3.1 Le principe de base . . . 75
5.3.2 La reproduction . . . 76
5.4 Architecture utilisée pour l'optimisation de l'agencement des caractères 77
5.4.1 Tranposition des algorithmes génétiques à l'agencement des caractères . . . 78
5.4.2 Potentiel d'amélioration . . . 78
5.4.3 Application de la reproduction . . . 80

6 Génération de claviers GAG : résultats et comparaisons **83**
6.1 Comparaison des performances avec les claviers de la langue anglaise . 83
6.1.1 Evaluation des algorithmes génétiques . . . 83
6.1.2 Application sur une disposition de touches optimisée . . . 86
6.2 Proposition d'un clavier optimisé pour le français . . . 87

6.3 Evolution des performances au cours des itérations 88

Conclusion **89**

III Claviers augmentés 91

Introduction **93**

7 Le système KeyGlass **99**

7.1 Clavier augmenté par *marking-menu* 99

7.1.1 Le principe 99

7.1.2 Les points positifs de la technique 100

7.1.3 Considérations pour une meilleure utilisation par des personnes handicapées 101

7.2 Le système KeyGlass 102

7.2.1 Le principe 102

7.3 Architecture du système de saisie 103

7.3.1 Un système modulaire 103

7.3.2 Mode de fonctionnement 104

7.3.3 Protocole de communication inter-module 104

7.4 Le système de prédiction de caractères utilisé 105

7.4.1 Description des systèmes testés 105

7.4.2 Evaluation de ces systèmes 106

7.4.3 Analyse des résultats 106

7.4.4 Le système de prédiction final 108

7.4.5 Le nombre de caractères proposés 109

8 Méthode de conception centrée utilisateur **111**
8.1 Première itération 111
8.1.1 Description du principe d'affichage des KeyGlasses 111
8.1.2 Evaluation théorique 112
8.1.3 Evaluation 115
8.1.4 Retours utilisateur 119
8.2 Seconde itération 120
8.2.1 Le système proposé 120
8.3 Evaluation du système 121
8.3.1 Objectifs et hypothèses 122
8.3.2 Protocole expérimental 122
8.3.3 Analyse des résultats 125
8.4 Discussion 135

Perspectives **137**

Conclusion et perspectives **139**

Conclusion **141**

Perspectives **143**

Annexes **147**

A Protocole de communication de la plate-forme E-Assiste **149**

A.5 Types de messages autorisés 149

A.5.1 Début d'une évaluation 149

A.5.2 Début d'un exercice 150

A.5.3 Données à recopier 150

A.5.4 Caractère saisi 150

A.5.5 Evenements du dispositif de pointage 150

A.5.6 Touche pressée 151

A.5.7 Prédiction de caractères 152

A.5.8 Gestion des KeyGlasses 152

A.5.9 Modification de variable 153

A.5.10 Initialisation de l'horloge 153

A.6 DTD du langage KTML 153

B Outils de la plate-forme E-Assiste **157**

B.7 Outils pour la partie sujet 157

B.7.1 Interpréteur de fichier 157

B.7.2 Interface de recopie de mots 158

B.7.3 Bandeau de défilement de texte 159

B.8 Outils pour la partie sauvegarde et analyse 159

B.8.1 Gestionnaire des traces temps réel 159

B.8.2 Analyseur des principales variables 160

B.8.3 Affichage de courbes et histogramme 160

B.8.4 Rejeu 161

B.9 Document de définition pour le format de description des claviers logiciels 162

Bibliographie **167**

Résumé **180**

Table des figures

1 Clavier logiciel Microsoft 3

2 Boucle perception-action pour l'interaction Homme-Machine (figure reprise de [Accot 01]) 9

3 Expérimentations menées par Fitts : A) Expérience "mouvements va-et-vient" [Fitts 54] ; B) Expérience "atteinte d'une cible" [Fitts 64] 13
4 Comment calculer la taille de la cible dans un pointage en deux dimensions ? 20
5 Interface utilisée pour réaliser la table d'actions pour des claviers comprenant 5 lignes et 6 colonnes (emprunté à [Hughes 02]) 22

6 Interface de présentation du texte à recopier (repris de [Matias 96]) 27

7 Architecture de la plate-forme E-Assiste 42
8 E-Assiste : une plate-forme distribuée 43

9 Clavier téléphonique classique 52
10 Clavier Twiddler 55
11 Le clavier Half-Qwerty : partie du clavier pour une utilisation, A) avec la main gauche ; B) avec la main droite 55
12 Le clavier DotNote : A) Interface contenant les caractères les plus fréquents ; B) Interface avec les caractères les moins utilisés. 56
13 Claviers téléphonique testé dans [MacKenzie 99b] : A) Clavier téléphonique classique ; B) Clavier JustType. 57
14 Comparaison des résultats Azerty vs. clavier téléphonique sur 2 populations : valides et handicapés moteur 58
15 Claviers alphabétiques : A) Configuration 5×6 proposée par [Lewis 99b] ; B) Configuration 3×13 proposée par [MacKenzie 99b]. 60
16 Claviers réalisés de manière intuitive : A) Clavier OPTI ; B) Clavier FITALY. 62
17 Claviers réalisés par techniques inspirées de la physique : A) Clavier Hook ; B) Clavier Metropolis. 65

18 Grilles testées pour l'optimisation 71
19 Grilles obtenues après optimisation 72
20 Représentation du problème sous forme d'arbre de solutions 74
21 Principe d'un algorithme génétique (repris de [Dréo 03]) 76

22 Architecture utilisée pour l'optimisation de l'agencement des caractères . . 77
23 Représentation de notre problème pour l'application des algorithmes génétiques : a) Chaque touche (définie par sa forme, sa taille et sa position sur le clavier) est affectée à une position dans la liste ; b) Exemple d'individu et de sa transposition sous forme de clavier. 79
24 Les différentes étapes de la reproduction : a et b) Reproduction ; c) Mutation ; d) Individu obtenu à la fin de la reproduction. 81

25 Claviers générés par algorithmes génétiques pour l'anglais 84
26 GAG III : Clavier généré par algorithme génétique pour l'anglais avec une disposition de touche plus compacte 86
27 Clavier généré par algorithme génétique pour le français 87
28 Evolution du meilleur individu au cours des itérations 88

29 Disposition de touches de différentes tailles 90

30 KeyStrokes : Exemple de clavier augmenté par une liste de mots les plus probables [Bérard 04a] 93
31 Exemple de clavier augmenté par une liste de mots prédits : le clavier POBox [Masui 98] 95
32 Le clavier Sybille : B. ré-organisation des caractères après la sélection du caractère 'c' (Figure A.) 96

33 Exemple du *marking-menu* sur un clavier logiciel 100
34 *Pie-menu* utilisé dans [Isokoski 04b] 100
35 Quatre KeyGlasses positionnées après la saisie du caractère 'b' 102
36 Architecture du système KeyGlass 103
37 Arbre lexicographique construit à partir des mots a, ami, amer, bon, bonne, bonjour, boire, boite. 105
38 Taux de prédiction correcte en fonction du système de prédiction et de l'avancée dans le mot 107

39 Designation des positions des KeyGlass 112
40 Quatre KeyGlasses positionnées après la saisie du caractère 'b' 112
41 Distances nécessaire en moyenne pour réaliser la tâche de saisie 117
42 Temps nécéssaire en moyenne pour réaliser la tache de saisie 118
43 Design du clavier pour la version 2 des KeyGlasses : A. clavier simple ; B. KeyGlasses qui apparaissent après la saisie du 's' ; C. KeyGlasses qui apparaissent après la saisie du 'e' qui était placé sur une KeyGlass. 120
44 Clavier SEB : clavier de base utilisé pour l'expérimentation 123
45 Taux d'utilisation des KeyGlasses 127
46 Distance 127
47 Durée 128
48 Erreur 129
49 Temps Réaction 130
50 Prédiction par session 131

51 Distance par session 132
52 Exemple de tracés ayant entrainé une augmentation de la distance 132
53 Vitesse par session : A. pour le clavier SEB ; B. avec le système KeyGlass . 133
54 Temps de réaction de chaque clavier 134
55 Temps de réaction spécifique sur le clavier SEB associé au système KeyGlass 134

56 Le principe Drag-and-Pop : A. Principe proposé par Baudisch et al. pour un bureau de PC [Baudisch 03] ; B. Principe appliqué à un clavier logiciel. Par exemple, après la saisie d'un 'b', on peut imaginer rapprocher les caractères 'a' et 'i'. 143
57 Extension de cibles : Plus le pointeur approche, plus la cible grossie. 144
58 Le clavier à touches extensibles : A. clavier réduit sans que le pointeur ne se trouve dessus ; B. Clavier déformé par la présence du pointeur à proximité du 'D' 145

59 Exemple de clavier généré à partir d'un fichier XML 157
60 Interface pour les exercices de recopie de mot. 158
61 Bandeau de défilement du texte à saisir. 159
62 Interface de recueil des traces des diverses expérimentations en cours. . . . 160
63 Interface de présentation des principaux résultats d'analyse 161
64 Courbe de vitesse du pointeur lors de la saisie. 161
65 Histogramme représentant les temps de pression sur les touches. 162
66 Interface de rejeu. 162

Liste des tableaux

1 Constantes de la loi de Fitts en fonction du dispositif (repris de [MacKenzie 91b]) 19
2 Constantes de la loi de Fitts en fonction de la façon de calculer la taille de la cible (Tableau repris de [MacKenzie 92b]) 21

3 Liste des mots choisis pour nos expérimentations 34

4 Comparaison des taux d'apparition des co-occurences les plus fréquentes dans les langues anglaise et française. 50

5 Extrait du tableau de [MacKenzie 02d] donnant les vitesses théoriques pour les claviers présentés dans ce manuscrit 63

6 Distances moyennes des différentes dispositions de touches 72

7 Performances des différents claviers optimisés pour l'anglais. 85
8 Performances des différents claviers optimisés pour l'anglais. 87

9 Résultat des simulations avec différents systèmes de prédiction 107
10 Performance du système de prédiction en fonction de la pondération de l'arbre lexicographique et du bigramme 108
11 Performance du système de prédiction en fonction du nombre de KeyGlass disponibles 109

12 Résultat en distance et temps des simulations selon les classes d'utilisateur 114
13 Ordre d'éxecution des exercices en fonction du groupe 116
14 Résultat de l'expérimentation avec un sujet handicapé 119
15 Résultats généraux de l'expérimentation de la seconde itération 126

Introduction

Depuis l'émergence du web dans les années 1990, l'informatique est devenue un formidable vecteur de communication utilisable par le plus grand nombre d'entre nous. Les lois successives du gouvernement français[1] et européen autour de "l'administration électronique" (*e-administration*) permettent l'accessibilité et l'accès à de nombreux services, notamment pour les personnes handicapées. Cependant, un des problèmes prégnants pour les personnes handicapées moteurs des membres supérieurs demeure le problème de la saisie de l'information sur ordinateur.

Avant d'exposer les problèmes liés à la saisie, relatons un bref historique sur l'évolution de celle-ci. Depuis l'invention de la machine à écrire, la saisie de texte au moyen de systèmes a connu trois grandes périodes [Soukoreff 04] : une première de la fin du XIX^e siècle au début du XX^e avec l'émergence de la machine à écrire et l'implantation de la disposition de touches QWERTY [Yamada 80] ; La seconde coïncide avec l'apparition de l'ordinateur comme outil de travail dans les années 1970-80 ; Enfin, nous nous trouvons actuellement, et ce depuis une dizaine d'années, dans la troisième ère de la saisie de texte avec l'apparition des dispositifs de petite taille et la saisie en mobilité.

Bien qu'il soit difficile de classer clairement les différentes avancées technologiques en matière de saisie de texte [Poirier 04], je propose de distinguer quatre grandes classes :

- **Les claviers physiques,** ce sont ceux que l'on trouve comme périphériques d'entrée sur tous les ordinateurs (de bureau ou portables). Les plus connus (et utilisés) sont ceux qui étaient déjà présents sur la machine à écrire, et qui ont été transposés aux claviers d'ordinateur : les claviers AZERTY et QWERTY.

- **Les claviers logiciels,** encore appelés claviers virtuels, sont une reproduction à l'écran d'un clavier physique (cf. Figure 1). La saisie de caractères se fait via un dispositif de pointage tel qu'un stylet, une souris, etc.

[1] Loi n°2005-102 du 11 février 2005 : "l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées", parue au Journal Officiel, Numéro 36 du 12 février 2005, page 2353

- **La saisie de texte gestuelle,** est une technique généralement utilisée sur les dispositifs mobiles munis d'un écran tactile au moyen d'une interaction au stylet. Elle peut être, elle-même, décomposée en deux grands ensembles :
  - La saisie gestuelle symbolique (ou analogique), basée sur des techniques de reconnaissance de geste et sur l'analogie entre le geste effectué et le caractère à saisir. La méthode la plus utilisée est *Unistroke* qui consiste à représenter chaque caractère par un tracé unique (l'exemple le plus connu étant l'alphabet *Graffiti*) ;
  - La saisie gestuelle non-analogique, basée sur le pointage de cible par geste : citons, par exemple, QuickWriting [Perlin 98], T-Cube [Venolia 94] ou encore Dasher [Ward 00]. La saisie de mots complets se fait alors en laissant le stylet en contact avec l'écran.
- **La reconnaissance vocale** qui transcrit un message vocal en une chaîne de caractères grâce au système de dictée vocale. Malgré les avancées qui ont été faites dans ce domaine pour rendre plus utilisable et robuste les systèmes de reconnaissance de la parole, leur efficacité pour saisir du texte correctement et rapidement reste à démontrer [Zhai 02a].

Notons que ces classes ne sont pas complètement indépendantes les unes des autres. Il est possible d'associer deux techniques de classes différentes pour optimiser la saisie. Prenons, l'exemple du système SHARK [Kristensson 04] : celui-ci est réalisé à partir d'un clavier logiciel optimisé sur lequel la saisie de caractères se fait au moyen d'un geste continu du premier au dernier caractère d'un mot. Ce système est ainsi à l'intersection des méthodes de claviers logiciels et de l'entrée de texte gestuelle.

Nous venons de présenter une catégorisation des systèmes qui permettent la saisie de texte directe. Mentionnons qu'il existe aussi des systèmes qui comportent des systèmes de prédiction de mots ou de caractères qui permettent d'accélérer la saisie. Ces systèmes de prédiction peuvent être associés à chacune des classes que nous venons de présenter.

Dans ce manuscrit, la problématique de recherche concerne la conception et la mesure de l'utilisabilité de claviers logiciels intégrant des composants d'interaction pour les personnes handicapées des membres supérieurs sans trouble cognitif. L'utilisation de périphériques de pointage adaptés (par exemple, trackball, eye-tracking [Majaranta 02, Hansen 04], ou encore mouvements de la tête [LoPresti 00, LoPresti 03]) en tant que mode de commande d'un clavier logiciel permet ainsi l'utilisation de l'ordinateur et la communication inter-personnelle.

A l'origine, les claviers AZERTY et QWERTY ont été conçus afin de ralentir la vitesse de saisie de texte sur les machines à écrire. En effet, pour éviter que les marteaux qui imprimaient les caractères ne s'entrecroisent trop, les caractères qui ont le plus de chance de se

succéder[2] avaient été éloignés. Aujourd'hui les ordinateurs ont en grande partie remplacé les machines à écrire et les contraintes mécaniques ont disparu. Pourtant la disposition de touches AZERTY (ou QWERTY) reste la plus utilisée sur les claviers d'ordinateur.

FIG. 1 – Clavier logiciel Microsoft

Plus étonnant encore, la plupart des claviers logiciels ont aussi adopté cette disposition de touches et de caractères (cf Figure 1). A la différence du clavier physique où l'on peut potentiellement utiliser les dix doigts pour saisir, le curseur du dispositif de pointage étant le seul moyen d'interaction avec le clavier logiciel, un utilisateur est obligé de réaliser tous les déplacements avec son dispositif de pointage. Les caractères les plus fréquemment utilisés n'étant pas voisins, l'amplitude du mouvement à réaliser entre deux caractères à saisir peut être importante (par exemple, du 'm' au 'a' pour le possessif 'ma'). L'un des principaux problèmes d'utilisabilité rencontré par les handicapés moteur des membres supérieurs réside dans la fatigue engendrée par la saisie de textes longs. Cette fatigue est due aux mouvements du curseur du dispositif à réaliser pour saisir du texte [Bérard 04a, Vella 05].

C'est pourquoi, nous souhaitons proposer au travers de ce manuscrit des solutions qui permettraient d'améliorer l'utilisabilité de ces systèmes pour les personnes handicapées moteur. Nous reprenons comme hypothèses pour la conception de nos systèmes celles de [Bérard 04a], c'est-à-dire que les utilisateurs potentiellement visés :

1. n'ont aucun accès à un clavier physique ;
2. peuvent utiliser un dispositif de pointage sur toute la surface de l'écran ;
3. ont accès au clic du dispositif de pointage ;
4. sont capables d'une précision normale ;

[2]La fréquence d'apparition d'un caractère après un autre est variable d'une langue à une autre : c'est pour cela qu'il existe un clavier AZERTY et un clavier QWERTY (respectivement pour le français et l'anglais).

5. mais sont sujets à une grande fatigabilité musculaire et parfois oculaire.

De nombreux travaux de recherche sur les claviers logiciels ont été conduits sur la langue anglaise [MacKenzie 02d, Zhai 02a], en proposant des méthodes d'agencement des touches et d'affectation de caractères à celles-ci à partir d'analyse de corpus. Pour l'essentiel, ces travaux ont proposé des claviers logiciels pour supports mobiles. Notre population cible d'utilisateurs finaux étant les personnes handicapées, la question de la minimisation de l'amplitude du mouvement est essentielle. De tels travaux n'existent pas à notre connaissance sur la langue française. C'est pourquoi, la problématique de cette thèse est de concevoir des agencements de touches/caractères optimaux pour la langue française et de les évaluer pour cette population tant de manière théorique qu'expérimentale.

Par conséquent, nous allons étudier et approfondir deux axes de recherche qui permettent de réduire les distances : d'une part, réorganiser les caractères sur le clavier logiciel pour les positionner de manière à diminuer les distances entre ceux que l'on saisit fréquemment, et d'autre part, proposer un système d'ajout dynamique de touches au cours de la saisie en y associant les caractères qui ont le plus de chance de succéder à ce qui vient d'être saisi.

Ces deux études indépendantes l'une de l'autre ont été menées en parallèle. L'organisation dans ce manuscrit n'est par conséquent pas à considérer comme une suite chronologique. Cette thèse s'est également préoccupée de concevoir une plate-forme d'évaluation en raison des besoins des projets par lesquels elle a été financée et des besoins des membres de l'équipe DIAMANT sur le thème de la "Communication Ecrite".

Organisation du manuscrit

Il s'ensuit l'organisation du manuscrit autour de trois parties :

1. Avant de discuter des optimisations que nous proposons pour les claviers logiciels, nous étudierons les méthodes qui nous sont offertes pour évaluer ce type de claviers logiciels. En effet, pour que l'on puisse parler d'optimisation, il faut disposer des moyens d'évaluer les performances que pourrait avoir un utilisateur avec notre système et ainsi les comparer avec les performances d'autres systèmes.

   Nous commencerons cette première partie par une étude sur les évaluations théoriques qui ne font pas intervenir d'utilisateurs finaux (Chapitre 1). Puis nous présenterons les différentes méthodologies d'expérimentation et les métriques qui sont utilisées pour faire une évaluation avec de potentiels utilisateurs finaux. Enfin, nous décrirons la plate-forme E-ASSISTE, environnement que nous avons réalisé pour

pallier le manque d'un référentiel commun permettant de faire des comparaisons entre divers claviers logiciels.

2. Nous étudierons, en seconde partie, les techniques d'optimisation utilisées pour proposer de nouvelles dispositions de touches et de caractères. L'objectif visé dans cette partie est de proposer des procédés afin de diminuer la distance moyenne à parcourir entre deux caractères saisis consécutivement. Nous procèderons ensuite à un état de l'art de ce qui a déjà été testé pour améliorer la disposition des caractères sur les claviers physiques et logiciels. Puis nous proposerons d'appliquer un algorithme de résolution de problème complexe (les algorithmes génétiques) au problème de recherche de la meilleure disposition. Enfin, nous présenterons les résultats obtenus avec notre méthode et nous les comparerons aux résultats des claviers optimisés existants.
3. Enfin, nous verrons qu'il est possible de diminuer les distances en apportant des comportements dynamiques au clavier logiciel. Nous commencerons cette troisième partie par une brève présentation des diverses manières "d'augmenter" un clavier logiciel, puis nous décrirons les motivations qui nous ont poussé à réaliser un nouveau système (appelé KeyGlass), ainsi que le principe et les caractéristiques de ce dernier. Enfin nous présenterons au travers de son cycle de conception et d'évaluation, comment nous avons réalisé et fait évoluer ce système de KeyGlass.

Première partie

Méthodologies d'évaluation des systèmes de saisie de texte

Introduction

Quelle que soit la tâche à réaliser avec un ordinateur, l'interaction entre l'homme et la machine peut être décomposée en différentes actions [Card 83] (cf. Figure 2) : L'homme conçoit une certaine information ①, puis la transmet à la machine ② qui la capture ③. Cette information est traitée par le système ④, puis la machine élabore à son tour une information en réponse ⑤ et la transmet à l'homme ⑥. Celui-ci la récupère ⑦ et l'analyse ⑧ et ainsi de suite.

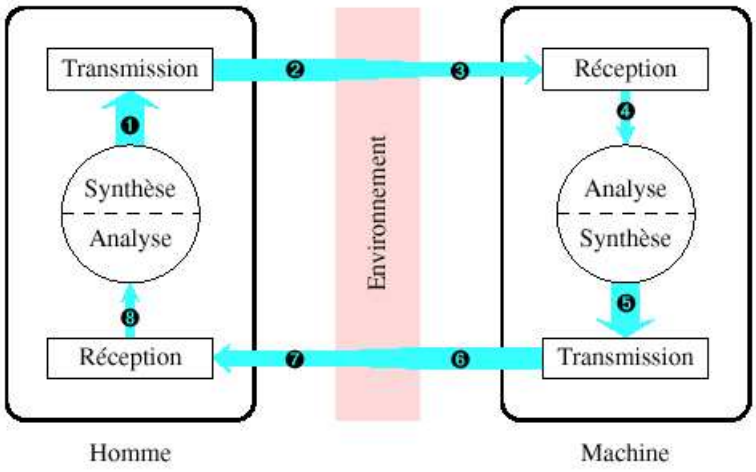


FIG. 2 – Boucle perception-action pour l'interaction Homme-Machine (figure reprise de [Accot 01])

La saisie de texte sur claviers logiciels est une activité du sujet qui peut être interprétée dans la boucle perception-action. L'information traitée par le système est la disposition des touches/caractères du clavier ⑤ qui sont affichés à l'écran ⑥, afin qu'ils soient perçus par l'utilisateur ⑦. Ce dernier identifie le caractère qu'il souhaite saisir ⑧, planifie le geste à faire pour saisir ce caractère ①, puis l'effectue ② au moyen d'un dispositif de pointage ③. Le système analyse alors le résultat de ce geste ④ pour voir si ce geste aboutit à la sélection d'une touche.

Lorsque l'on conçoit un clavier logiciel, il est assez facile de prévoir ce qui va se passer du côté de la machine : le temps qu'il va lui falloir pour analyser les informations provenant de l'utilisateur ainsi que celui nécessaire pour s'adapter et renvoyer de nouvelles informations si cela est nécessaire. Par contre, il est plus complexe de comprendre la réaction de l'utilisateur et quel temps il lui faut pour réaliser une tâche de saisie. L'évaluation d'un clavier logiciel consiste généralement à déterminer le temps mis par un utilisateur pour analyser l'information (chercher visuellement la touche) ainsi que celui nécessaire à la sélection/validation de la touche.

Pour évaluer les performances de l'utilisateur, il existe deux méthodes :

- **Une évaluation *a priori*** qui, à partir de lois théoriques, déterminera quelles seront *a priori* les performances de l'utilisateur avec le clavier ;
- **Une expérimentation** de l'activité de saisie de texte par des sujets avec le clavier évalué ; celle-ci consiste à analyser *a posteriori* l'interaction entre l'utilisateur et le clavier à partir des données recueillies lors de l'expérimentation.

Nous présenterons et discuterons dans cette partie des avantages et inconvénients des deux méthodes, ainsi que des contextes dans lesquels il est intéressant d'utiliser l'une ou l'autre. De nombreuses mesures de performances, d'erreurs, etc. commencent à devenir des standards. Malheureusement, il n'existe pas de référentiel commun permettant de comparer facilement deux systèmes. Afin de pallier ce problème, et pour répondre à divers besoins d'évaluation de techniques d'interaction, nous avons spécifié, au sein de l'équipe DIAMANT, une plate-forme d'évaluation (nommée E-ASSISTE) que j'ai ensuite développé et adaptée aux diverses expérimentations. J'en décrirai les motivations, les principes généraux et la conception dans le dernier chapitre de cette partie.

1

Les lois prédictives

Nous entendons par loi prédictive, toute loi qui ne fait pas intervenir d'utilisateurs finaux mais qui se base seulement sur un calcul *a priori* des performances que pourrait avoir un utilisateur avec le système à tester. Dans le cadre d'un clavier logiciel, ce calcul estime généralement la vitesse de saisie de texte, soit en nombre de caractères qu'il est possible de saisir par seconde, soit en nombre de mots par minute.

Une tâche de saisie de texte sur clavier logiciel engendre plusieurs actions de l'utilisateur : la recherche visuelle d'une touche, puis le geste de déplacement jusqu'à la touche et enfin la sélection (validation) de celle-ci.

Ces actions ont fait l'objet de nombreuses recherches en Interaction Homme-Machine et en psychologie expérimentale sur l'étude du mouvement [Guiard 04]. Chacune d'elles a fait l'objet de proposition de lois prédictives permettant d'estimer le temps nécessaire pour l'accomplir.

Nous commencerons par présenter ces lois, puis nous montrerons comment en intégrant celles-ci, différents modèles prédisent la vitesse théorique de saisie de texte sur un clavier logiciel. Nous finirons par un bilan sur ces différents modèles en étudiant dans quel contexte d'évaluation il est intéressant ou non de les utiliser.

1.1 Temps de recherche visuelle d'une cible

Le temps de recherche visuelle d'une cible correspond au temps de réaction nécessaire pour amorcer l'action qui suit. Une relation permettant de déterminer ce temps a été proposée indépendamment par Hick [Hick 52] en 1952 et Hyman [Hyman 53] en 1953. En s'appuyant sur les théories de l'information [Shannon 49], ils montrèrent que ce temps de réaction était proportionnel au nombre de cibles qu'il était possible de choisir. Ce temps de réaction est au moins équivalent au temps de reconnaissance d'une cible, qui lui même ne

peut pas être inférieur au temps nécessaire aux systèmes sensoriels pour qu'ils fournissent la quantité d'information. Celle-ci est équivalente au nombre de cibles qu'il est possible de choisir.

Hick proposa alors la loi (généralement appelée loi de Hick-Hyman) suivante :

$$TR = a + b \times \log_2(N) \quad (1)$$

où TR représente le temps de recherche d'une cible parmi les N qu'il est possible de choisir ; a et b sont des constantes qui sont calculées empiriquement par régression linéaire : b étant la pente de la droite de régression et a l'ordonnée à l'origine de cette même droite. Ces constantes sont dépendantes du contexte d'utilisation. La courbe de régression linéaire est calculée à partir d'un ensemble de points où l'axe des abscisses et celui des ordonnées représentent respectivement le nombre de cibles et le temps nécessaire pour en rechercher une. L'ensemble des points est obtenu après expérimentations avec plusieurs sujets qui effectuent diverses tâches de recherche de cibles avec un nombre de cibles différent lors de chaque tâche. Nous discuterons de la valeur à leur donner par la suite dans le cas précis de la tâche de saisie de texte (cf. section 1.3.3).

1.2 Temps de pointage d'une cible

1.2.1 La loi de Fitts

C'est Woodworth [Woodworth 99] à la fin du XIX^e siècle (1899), qui fut le pionnier dans ce domaine. Il démontra qu'il était incompatible d'associer à la fois une grande rapidité du mouvement et une haute précision dans le pointage d'une cible.

Ce n'est qu'en 1954 que Fitts [Fitts 54] proposa une loi mettant en relation ces deux contraintes : la rapidité du mouvement et la précision du pointage. Il se basa aussi sur la théorie de l'information et plus particulièrement sur la formulation de Shannon [Shannon 49]. Celle-ci propose que pour un signal D, avec du bruit W, la capacité C de transmission du message soit :

$$C = \log_2(\frac{D+W}{W}) = \log_2(\frac{D}{W}+1) \quad (2)$$

Fitts transposa cette formule au problème de la transmission des messages dans le système perceptuo-moteur. Il définit ainsi l'indice de difficulté (ID) à transmettre le message de l'émetteur au récepteur permettant de réaliser un mouvement de distance D,

où le bruit représente la précision avec laquelle ce geste doit être effectué pour atteindre la cible de largeur W. Cet indice de difficulté proposé par Fitts était :

$$ID = \log_2(\frac{2D}{W}) \quad (3)$$

A partir de trois expériences réalisées pour vérifier sa théorie, Fitts établit la loi (qui porte son nom) permettant de prédire le temps nécessaire pour pointer une cible de largeur W située à une distance D :

$$T = a + b \times ID = a + b \times \log_2(\frac{2D}{W}) \quad (4)$$

où :

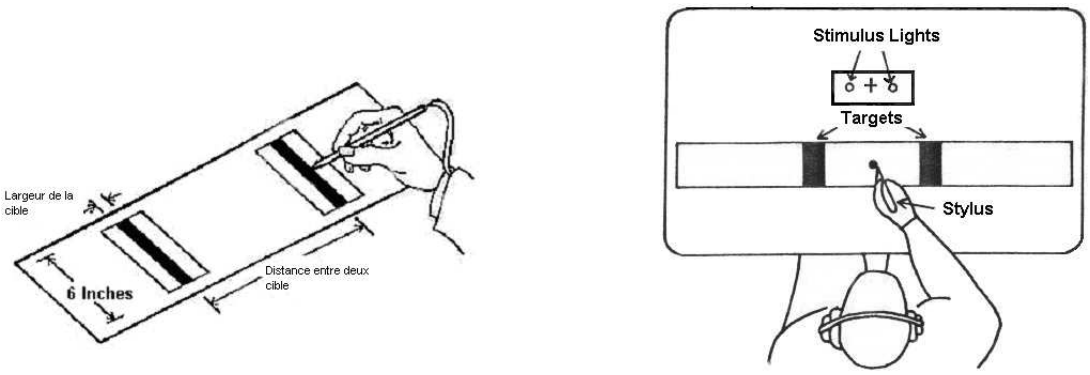
- T est exprimé en secondes ;
- D représente l'amplitude du mouvement à effectuer ;
- W la taille de la cible à atteindre ;
- a et b sont des constantes dont les valeurs sont obtenues de la même manière que celles de la loi de Hick-Hyman : c'est-à-dire par régression linéaire. L'axe des abscisses et l'axe des ordonnées y représentent respectivement l'indice de difficulté (ID) et le temps de pointage d'une cible. La valeur à leur affecter sera également discutée à la section 1.3.3.

1.2.2 Ajustement de la loi de Fitts pour une utilisation en Interaction Homme-Machine

Les premières expérimentations de Fitts consistèrent à effectuer un mouvement de va et vient entre deux cibles (cf. Figure 3 A.). Cette action de va et vient n'étant que très peu réalisée naturellement, il démontra alors avec Peterson [Fitts 64] en 1964 que sa première loi était aussi applicable dans le cas d'un mouvement discret, comme l'atteinte d'une cible fixe (cf. Figure 3 B.).

Lorsque Fitts établit sa loi, il se basa sur des expérimentations faites sur des supports où la cible à atteindre était physique (cf. Figure 3). Ce n'est qu'à la fin des années 70 que Card et ses collègues [Card 78] montrèrent que le temps nécessaire pour pointer une cible (virtuelle) sur un ordinateur pouvait être modélisé par la loi de Fitts[3]. Ils démontrèrent

[3]Cette modélisation est possible malgré l'intermédiaire qui se trouve entre le geste effectué avec un dispositif de pointage et la cible virtuelle qui est affichée à l'écran.



A) B)

FIG. 3 – Expérimentations menées par Fitts : A) Expérience "mouvements va-et-vient" [Fitts 54]; B) Expérience "atteinte d'une cible" [Fitts 64]

aussi[4] que le pointage via une souris (ou un autre dispositif de pointage) sur un ordinateur avait des propriétés similaires au pointage physique.

Malgré de nombreuses propositions[5] faites pour améliorer la loi de Fitts, celle-ci reste encore la plus utilisée en psychologie expérimentale. Les chercheurs en Interaction Homme-Machine ont eux aussi adopté depuis quelques années une formulation de la loi de Fitts pour prédire le temps de déplacement vers une cible à l'écran. Généralement reconnue comme n'étant pas optimale et robuste à toutes les situations, la loi de Fitts initiale (cf. équation 4) fut légèrement modifiée [MacKenzie 92a] pour parvenir à une formulation plus proche de la théorie de Shannon. Le temps de pointage d'une cible est généralement établi par :

$$T = a + b \times \log_2(\frac{D}{W} + 1) \quad (5)$$

où D, W, a et b sont les mêmes variables que dans l'équation 4.

1.3 Prédiction des performances d'un clavier

Landauer et Nachbar [Landauer 85] montrèrent que les lois de Hick-Hyman et de Fitts pouvaient servir à prédire le temps nécessaire pour effectuer, sur ordinateur, une tâche de "mouvement plus sélection", respectivement pour le choix de la cible à atteindre (souvent un composant graphique) puis le pointage de celle-ci.

[4] Ainsi que d'autres chercheurs par la suite [MacKenzie 92a, Douglas 97].

[5] Le lecteur pourra se référer à [Meyer 90] et [Plamondon 97] pour avoir une revue de ces propositions.

La saisie de texte sur clavier logiciel se définissant comme une double tâche de "mouvements plus sélection", des modèles ont été proposés autour de ces lois. Les plus connus sont : le Modèle Keystroke-Level Model basé sur une évaluation à partir de scénarii, et le modèle de Soukoreff et Mackenzie qui utilise des connaissances linguistiques pour évaluer la vitesse théorique d'un clavier logiciel.

1.3.1 Le modèle Keystroke-Level Model

Le modèle Keystroke-Level Model (KLM) proposé par Card, Moran et Newell au début des années 80 [Card 80, Card 83], a été développé pour prédire le temps nécessaire pour réaliser une tâche particulière sur un ordinateur. Des applications de ce modèle ont été proposées [Koester 94, Dunlop 00] dans le domaine de la saisie de texte. Sa prédiction du temps (noté T_{EXEC}) est basée sur l'addition des temps estimés pour réaliser six actions de base :

- t_K : Le temps de frappe ;
- t_P : Le temps de pointage ;
- t_H : Le temps de déplacement de la main ;
- t_D : Le temps de dessin ;
- t_M : Le temps relatif à l'activité mentale ;
- t_R : Le temps de réponse de la machine.

$$T_{EXEC} = t_K + t_P + t_H + t_D + t_M + t_R \tag{6}$$

Selon le scénario de la tâche à accomplir, certains de ces temps peuvent être utilisés à plusieurs reprises et d'autres ignorés. Par exemple, dans [Dunlop 00], l'auteur ne prend en compte que trois de ces temps : t_K,t_H et t_M. A partir de ceux-ci, il détermine la vitesse de saisie sur un clavier téléphonique. Les lois de Fitts et Hick-Hyman décrites précédemment peuvent être utilisées pour prédire le temps de certaines sous actions.

Nous soulignons la difficulté d'effectuer une évaluation convenable d'un clavier avec ce modèle. Ceci implique de proposer la saisie d'un texte qui représente les fréquences d'apparition des bi grammes de la langue utilisée. Cet exercice de création de texte est complexe et nécessite de collaborer avec des spécialistes du Traitement Automatique de la Parole.

1.3.2 Le modèle de Soukoreff et Mackenzie

C'est pour éviter ces contraintes de création de texte, que Soukoreff et MacKenzie [Soukoreff 95] proposèrent, en 1995, une formule plus globale permettant de prédire *a priori* les performances de saisie d'un utilisateur avec un clavier logiciel à partir des seules lois de Fitts et Hick-Hyman et de connaissances linguistiques statistiques. Cette évaluation purement théorique prend en compte la disposition des touches sur le clavier (c'est à dire principalement leur position et leur taille), l'agencement des caractères sur ces touches et une table statistique des fréquences des bi grammes de caractères de la langue pour laquelle ce clavier est conçu.

Cette loi repose sur l'équation 5 proposée par Mackenzie pour prédire le temps de déplacement nécessaire pour pointer une cible donnée :

$$T_{ij} = a + b \times \log_2 \left(\frac{D_{ij}}{W} + 1 \right) \tag{7}$$

Avec :

- T_{ij} le temps nécessaire pour sélectionner[6] la touche t_j à partir de la touche t_i ;
- La distance D_{ij} qui sépare la touche associée au caractère $C_i \in \mathcal{A}$ (avec $\mathcal{A}$ l'alphabet des caractères présents sur le clavier logiciel) sur laquelle on se trouve et la touche associée au caractère $C_j \in \mathcal{A}$ correspondant à celle où l'on veut aller ;
- La largeur W de la touche que l'on souhaite atteindre.

Pour évaluer un clavier complet, il faut ainsi prendre en compte la sélection d'une même touche deux fois consécutivement. Dans la loi initialement proposée par Fitts, la prédiction du temps de pointage d'une cible est validée dans le cas d'une distance positive non nulle. Comme une double sélection prend toujours un temps minimum, le temps pour saisir un caractère est formulé de la manière suivante :

$$T_{ij} = \begin{cases} a + b \times \log_2 \left(\frac{D_{ij}}{W} + 1 \right) & si\ i \neq j \\ T_{repeat} & si\ i = j \end{cases} \tag{8}$$

Il s'en suit la formule du temps moyen (exprimé en seconde par caractère) pour sélectionner un caractère sur un clavier logiciel :

[6] Nous parlons ici de sélection : ceci implique l'action de pointage de la touche, mais aussi le temps nécessaire pour la valider par un clic par exemple.

$$\overline{MT} = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{A}} (P_{ij} \times T_{ij}) \quad (9)$$

Avec P_{ij} la probabilité que le caractère $C_i \in \mathcal{A}$ soit suivi du caractère $C_j \in \mathcal{A}$, et :

$$\sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{A}} P_{ij} = 1 \quad (10)$$

En règle générale, sachant que la majorité des recherches actuelles ont été menées dans des pays anglo-saxons, et qu'ils n'utilisent que très peu les accents, l'évaluation a porté sur des claviers limités aux 26 lettres de l'alphabet latin plus le caractère espace, soit un alphabet de 27 caractères.

Cependant, la formule 9, ne donne que le temps moyen nécessaire pour se déplacer d'un caractère à un autre sur un clavier logiciel. Voyons maintenant comment calculer à partir de cette première formule la vitesse théorique de saisie de texte d'un utilisateur avec ce clavier logiciel en fonction des profils utilisateur.

Nous distinguons deux profils d'utilisateur de clavier :

- **Les experts,** ceux qui ont l'habitude d'utiliser le clavier donné et dont on considère qu'ils connaissent bien la disposition des caractères. Ceci a pour conséquence que le temps de recherche visuelle d'un caractère est alors considéré comme nul ;
- **Les novices,** ceux qui connaissent mal la disposition des caractères sur le clavier. Le temps de recherche d'une touche peut alors être important pour ces utilisateurs.

Ces types d'utilisateurs sont les deux cas extrêmes. Il existe une majorité de personnes se situant dans l'intervalle de ces deux catégories. Néanmoins, nous constatons de la lecture de la littérature que ce sont souvent ces seuls types de classification qui sont pris en considération lors de la prédiction des performances d'un clavier. Les performances de saisie des autres utilisateurs devraient normalement être comprises dans l'intervalle borné par les performances des novices et celles des experts.

Deux valeurs (proposées par Soukoreff et MacKenzie dans [Soukoreff 95]) sont déterminées pour connaître les performances *a priori* d'un utilisateur. Il s'agit du nombre de caractères saisis par seconde (en anglais *characters per second* (CPS)) et du nombre de mots qu'il est possible de saisir par minute (en anglais, *words per minute* (WPM)) en estimant qu'un mot est en moyenne composé de 5 caractères. Ceci est une estimation du nombre moyen de caractères par mots pour la langue anglaise [Gentner 83] pour laquelle il n'y a pas eu de distinction faite entre les lettres, les espaces et les signes de ponctuation.

Les performances minimales *a priori* d'un utilisateur avec un clavier

La limite inférieure correspond aux performances que l'on peut espérer d'un utilisateur qui utilise pour la première fois le clavier qui lui est proposé. Celles-ci sont calculées en fonction du temps de recherche visuelle d'une touche (TR) (cf. équation 1) et du temps de sélection de celle-ci ($\overline{MT}$). Nous obtenons ainsi pour la limite inférieure :

$$CPS_{min} = \frac{1}{\overline{MT} + TR} \tag{11}$$

$$WPM_{min} = \frac{CPS_{min} \times 60}{5} \tag{12}$$

Les performances maximales *a priori* d'un utilisateur avec un clavier

La limite supérieure donne les performances maximales qu'un utilisateur pourrait théoriquement atteindre[7] après de nombreuses heures passées à utiliser ce clavier[8]. Dans le cas d'un utilisateur expert, le temps de recherche étant estimé à nul ($TR = 0$), le nombre de caractères par seconde est de :

$$CPS_{max} = \frac{1}{\overline{MT}} \tag{13}$$

Nous obtenons alors :

$$WPM_{max} = \frac{CPS_{max} \times 60}{5} \tag{14}$$

1.3.3 Quelles valeurs pour les constantes pour une évaluation théorique de clavier ?

Comme nous avons pu le constater au travers des équations précédentes, chacune d'entre elles fait intervenir des constantes a et b. Ces constantes sont calculées en fonction du contexte d'utilisation et des sujets.

[7] Même si l'on considère qu'un utilisateur progresse indéfiniment [De Jong 57]

[8] [Cooper 83] avance, par exemple, le nombre de 100 heures pour obtenir des performances similaires avec le clavier DVORAK qu'avec le clavier QWERTY

1.3.3.1 Les constantes de la loi de Fitts

Les constantes a et b de la loi de Fitts sont assez dépendantes du contexte dans lequel on souhaite réaliser le pointage : le dispositif de pointage, le support, ou encore la tâche sont des facteurs qui peuvent influencer les valeurs à attribuer aux constantes. Le tableau 1 donne les valeurs des constantes, pour la même tâche de pointage, où seul le dispositif d'entrée varie.

Dispositif	a *(ms)*	b *(ms/bit)*	IP = 1/b *(bits/s)*
Souris	-107	223	4,5
Stylet	-55	204	4,9
Trackball	75	300	3,3

TAB. 1 – Constantes de la loi de Fitts en fonction du dispositif (repris de [MacKenzie 91b])

Nous constatons dans ce tableau que certaines valeurs de a sont négatives. Pour éviter d'avoir une valeur négative sur une tâche de pointage où l'indice de difficulté de celle-ci est très faible, les valeurs négatives de a sont généralement remplacées par $a = 0$.

Lors de l'évaluation du clavier Metropolis, Zhaï et ses collègues [Zhai 02b] ont réévalué la valeur de ces constantes à partir d'une tâche de saisie. Les données récoltées pour construire la courbe de régression linéaire, proviennent d'une expérimentation réalisée avec 12 sujets qui devaient saisir du texte sur le clavier Metropolis [Zhai 00] sur une tablette Wacom avec un stylet comme dispositif de pointage. Les valeurs obtenues sont respectivement : a=0,083 s, b=0,127 s/bit soit IP=7,9 bits/s.

Enfin, nous avons vu que pour l'évaluation d'un clavier, il est nécessaire d'attribuer une valeur dans le cas où l'on saisit consécutivement deux fois le même caractère. Cette valeur correspond au temps qu'il y a entre les deux clics, sachant que l'indice de difficulté vaut ici 0 (car la distance est nulle). Cette valeur n'a que peu d'importance sur le résultat, et dépend aussi du contexte (matériel) dans lequel on se trouve. Ainsi on retrouve plusieurs valeurs dans la littérature : 0,153 [Soukoreff 95], 0,135 [Zhang 98] ou 0,127 [MacKenzie 99a].

1.3.3.2 Les constantes a et b de la loi de Hick-Hyman

Les valeurs des constantes a et b proposées par Hick-Hyman [Hick 52, Hyman 53] n'ont pas été remises en cause. Welford [Welford 68] reprit en 1968 les mêmes constantes à savoir $a = 0$ et une valeur de $IP = 1/b$ comprise entre 5 et 7 bits par seconde.

Dans le cas de l'évaluation des claviers logiciels, comme nous cherchons la borne inférieure des performances théoriques d'un utilisateur, [Soukoreff 95] proposèrent de prendre la valeur la plus faible. Leur choix s'explique par le fait qu'il utilise la loi de Hick-Hyman pour prédire la vitesse de saisie de texte minimale que pourrait avoir un utilisateur. C'est pourquoi, ils ont pris la valeur la plus faible de IP pour prévoir le temps maximum qu'un utilisateur pourrait prendre pour chercher un caractère sur le clavier. Nous utiliserons comme valeur pour nos estimations : IP = 5 bits par seconde, soit b = 1/5 = 0.2 seconde par bit.

1.3.3.3 Le clavier : un problème de pointage en 2 dimensions

La loi de Fitts a été définie pour estimer le temps de pointage d'une cible en ne prenant en compte qu'une dimension : la tâche à réaliser se limitait à un déplacement horizontal entre deux cibles (cf. Figure 3). La taille de la cible était alors la largeur de celle-ci.

Dans le cas des claviers logiciels, la tâche de pointage se réalise en deux dimensions : un utilisateur désigne des cibles en réalisant un mouvement horizontal, vertical ou même diagonal. Plusieurs études [Card 78, Jagacinski 85] ont néanmoins utilisé la loi de Fitts prévue pour une dimension pour estimer des temps de pointage sur des tâches faisant intervenir deux dimensions.

Le principal problème dans le pointage d'une cible en deux dimensions est la prise en compte de la taille de la cible. Doit-on continuer à prendre la largeur de la cible ? Comment cela se passe-t-il dans le cas d'une cible asymétrique ? Plusieurs études sur la taille de la cible sont rapportées dans [Gillan 90, MacKenzie 91a, MacKenzie 92b] pour proposer des stratégies d'adaptation de la taille de la cible. Les recommandations qui ressortent sont :

- Dans le cas d'une cible rectangulaire, prendre le minimum entre la largeur (W) et la hauteur (H) de la cible ;
- Dans le cas d'une cible plus complexe, prendre la longueur du segment (W') représentant l'intersection entre le vecteur déplacement lors du mouvement (et passant par le centre de la cible) et la cible (cf. Figure 4).

La manière de calculer la taille de la cible implique aussi la ré-évaluation des constantes de la loi de Fitts. Le tableau 2 illustre les différences sur la valeur des constantes dues au procédé de calcul.

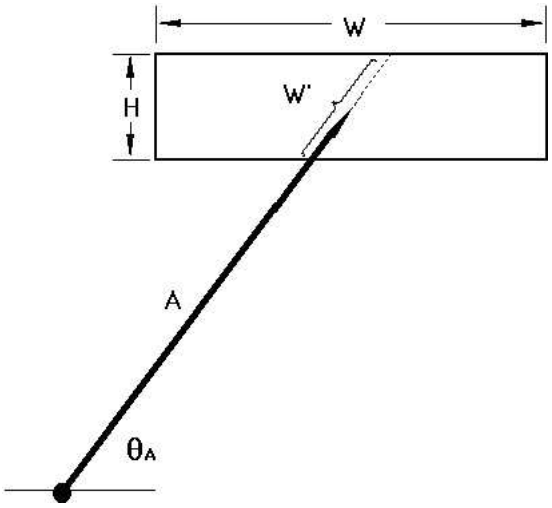


FIG. 4 – Comment calculer la taille de la cible dans un pointage en deux dimensions ?

Dispositif	a *(ms)*	b *(ms/bit)*	IP = 1/b *(bits/s)*
min(W,H)	230	166	6,0
W'	337	160	6,3
W+H	402	218	4,6
W × H	481	173	5,8
W	409	135	7,4

TAB. 2 – Constantes de la loi de Fitts en fonction de la façon de calculer la taille de la cible (Tableau repris de [MacKenzie 92b])

1.3.4 Bilan

Dans le domaine de l'évaluation de claviers logiciels, la loi de Soukoreff et Mackenzie [Soukoreff 95] fait maintenant office de référence. Cependant pour être fiable dans l'estimation des performances, cette loi nécessite une pré expérimentation afin de régler les constantes a et b de la loi de Fitts en fonction du contexte d'utilisation (mobilité, handicap ...) et du calcul de la taille des touches.

Même après avoir fixé ces constantes, la méthode n'est pas fiable à 100%. Selon une étude de Mackenzie [MacKenzie 01b], l'estimation des performances d'un utilisateur novice est trop élevée par rapport à celles réellement obtenues en situation. En effet, à partir d'une expérimentation réalisée avec 12 sujets mis en situation d'utilisateurs novices

(c'est-à-dire saisie sur un clavier dont ils ne connaissaient pas la disposition des caractères avant l'expérimentation), ils ont obtenu une vitesse de saisie de texte moyenne 40% moins élevée que celle estimée par les lois prédictives.

Néanmoins, cette loi peut être utilisée à but de comparaison de performances : si les claviers à comparer ne diffèrent que dans la position des touches et la disposition des caractères sur celles-ci, il est alors possible d'utiliser cette loi sans se soucier de la précision des constantes. En effet, pour une comparaison entre deux claviers, quelle que soit la valeur de la constante, si un système A a de meilleures performances qu'un système B, il aura toujours de meilleures performances avec d'autres valeurs pour ces constantes, et il est quasiment certain (du moins si l'écart de performances entre les deux systèmes est significatif) que les performances de l'utilisateur seront meilleures avec le système A qu'avec le B.

1.4 Entre loi prédictive et expérimentation : quelle solution ?

Comme nous venons de le voir, le modèle de Soukoreff et Mackenzie nécessite une pré expérimentation pour fixer les valeurs des constantes utilisées. Une fois que ces constantes sont figées, on peut utiliser ce modèle pour l'évaluation de tout clavier, en restant dans le même contexte d'utilisation (matériel, dispositif d'entrée, profil de sujet, ...) que celui de la pré expérimentation. Néanmoins, l'estimation des performances reste approximatives.

C'est pourquoi, Hughes et ses collègues [Hughes 02] ont proposé une méthode d'évaluation plus fine qui consiste à remplacer la prédiction du temps de déplacement d'une touche à une autre par un temps concret obtenu lors d'une expérimentation réalisée au préalable. Leur méthode consiste à prendre la même disposition de touches que celle du clavier à évaluer sans y associer de caractère dans un premier temps (cf. Figure 5). Lors de l'expérimentation, les sujets devaient réaliser tous les déplacements possibles d'une touche à une autre[9]. Cette première expérimentation permet alors d'associer un temps de déplacement à chaque couple de touches. L'ensemble de ces couples est sauvegardé sous la forme d'une table d'actions.

A partir de la disposition de touches correspondant au clavier, de la table d'action associée à cette disposition de touches et d'une table de bi gramme, on obtient alors une

[9]Les déplacements à réaliser sont demandés de manière aléatoire par le système pour éviter l'anticipation sur le mouvement suivant.

A5	**B5**	**C5**	**D5**	**E5**	**F5**
A4	**B4**	**C4**	**D4**	**E4**	**F4**
A3	**B3**	**C3**	**D3**	**E3**	**F3**
A2	**B2**	**C2**	**D2**	**E2**	**F2**
A1	**B1**	**C1**	**D1**	**E1**	**F1**

FIG. 5 – Interface utilisée pour réaliser la table d'actions pour des claviers comprenant 5 lignes et 6 colonnes (emprunté à [Hughes 02])

estimation plus fine des performances *a priori* d'un utilisateur avec le clavier évalué. Cette estimation du temps (en caractères par seconde) $R(K, P, E)$ est obtenue par :

$$R(K, P, E) = \frac{1}{\sum_{\alpha,\beta} P(\alpha, \beta) E(K(\alpha), K(\beta))} \tag{15}$$

où :

- $P(\alpha, \beta)$ représente la probabilité que le caractère β succède au caractère α ;
- la fonction $E(K(\alpha), K(\beta))$ représente le temps qu'il faut pour se déplacer du caractère α au caractère β sur le clavier ayant une disposition de touche K.

Cette méthode a l'avantage de proposer une évaluation théorique plus précise que celle de Soukoreff et MacKenzie. Elle est également moins coûteuse en temps que de réaliser plusieurs expérimentations avec des sujets pour tester différents claviers ayant la même disposition de touches mais pas le même agencement des caractères. Cependant, à la différence du modèle de Soukoreff et Mackenzie qui fonctionne pour différentes positions de touche, cette méthode doit être refaite à chaque fois que l'on change la position des touches.

1.5 Nos choix pour la suite

Nous pouvons voir l'utilisation des lois prédictives de deux manières :

- D'une part, comme une évaluation à part entière, qui débouche sur une mesure de performance qui doit refléter la valeur du clavier et ainsi connaître parfaitement ses limites. Dans ce cas, une pré expérimentation est nécessaire pour avoir un modèle le plus robuste possible par rapport au contexte d'utilisation. Si l'on connaît la

disposition des touches à l'avance et que l'on veuille tester différentes dispositions de caractères, le modèle de Hughes est le plus approprié pour avoir des valeurs précises. Dans le cas où la position des touches n'est pas encore figée, le modèle de Soukoreff et Mackenzie est plus approprié car il évite d'avoir à refaire une pré expérimentation pour toute nouvelle disposition de touches. Dans les deux cas, faire une pré expérimentation reste une tâche coûteuse, surtout lorsqu'on veut la faire faire à des personnes handicapées des membres supérieurs. De plus, pour que la pré expérimentation soit robuste en terme de données, il est nécessaire d'avoir un nombre de sujets assez conséquent (au moins une douzaine). Cela revient à faire une expérimentation classique (voir Chapitre 2) et donc les modèles prédictifs perdent leur principal intérêt qui est d'être peu coûteux en temps ;

- D'autre part, on peut utiliser ces modèles pour effectuer des comparaisons entre claviers. Dans ce cas, même si les valeurs ne sont pas très précises, cela n'a pas d'effet sur ce que l'on recherche, à savoir un classement des performances de claviers. Si l'on utilise les modèles prédictifs avec cet objectif, celui de Soukoreff et Mackenzie est alors le plus approprié. En effet, des valeurs pour les constantes ont déjà été fixées pour plusieurs types de contexte. Il est possible de prendre les valeurs les plus adéquates, et de faire les comparaisons entre claviers avec ces valeurs. Le modèle de Hughes n'est pas utilisable, car il nécessite une table d'actions. Celle-ci étant spécifique à une disposition de touches, il n'est pas évident de trouver cette table pour la disposition de touches souhaitée.

A partir de ce constat, nous avons choisi de n'utiliser ces lois prédictives et plus particulièrement le modèle de Soukoreff et Mackenzie que pour la comparaison de clavier. Celui-ci nous permettra de comparer notre approche (cf. Chapitre 5) aux autres déjà réalisées. Les conditions d'utilisation du modèle pour notre cas sont détaillées au Chapitre 6.

2

Méthodologies d'évaluation

Contrairement aux claviers physiques, les systèmes de saisie de texte logiciels ne se contentent généralement pas de proposer une disposition de caractères statique. Une particularité de ces systèmes est de proposer des fonctionnalités supplémentaires permettant d'accélerer un peu plus la vitesse de saisie de texte. On trouve par exemple des systèmes tels que Sybille qui réorganisent les caractères en fonction de la saisie [Schadle 03], ou qui proposent la fin des mots[10]. Ces systèmes que nous définissons comme 'dynamiques' font généralement intervenir un système de prédiction de lettres ou de mots. Même s'il est possible d'estimer les performances théoriques de ces types de système, il est préférable de les évaluer en faisant intervenir des utilisateurs finaux. En effet, les événements dynamiques qui se produisent sur le système de saisie de texte peuvent perturber l'utilisateur et provoquer alors une diminution de ses performances avec ce système, ou au contraire réduire l'effet de fatigue occulaire et musculaire pour une personne handicapée [Bérard 04a].

Mais dans ce genre d'expérimentation, que cherche-t-on à étudier ? Quelles sont les variables qui nous intéressent ? Comment peut-on procéder pour évaluer un système de saisie de texte afin que cette expérimentation soit une bonne estimation de ce qui passera lors d'une utilisation quotidienne et non contrainte ?

2.1 Les tâches à réaliser

Lors d'une évaluation avec des sujets, l'activité de saisie de texte peut consister en :

- une recopie d'un texte ou d'un ensemble de mots ;
- une saisie libre d'un texte de leur propre imagination.

[10] soit sous forme de liste [Darragh 90, Pévédic 97], soit de manière automatique [Boissière 90]

2.1.1 La tâche de recopie

2.1.1.1 La problématique du focus multiple d'attention

Le focus d'attention est représenté par le nombre de points où une attention visuelle peut être demandée à l'utilisateur par la tâche [MacKenzie 02d]. Par exemple, un utilisateur expert d'un clavier physique qui doit faire une tâche de recopie ne regardera que le texte source à recopier : on parle alors de simple focus d'attention. Dans le cas de l'utilisation d'un clavier logiciel avec un stylet, pour la même tâche de recopie, on parle de double focus d'attention car l'utilisateur est obligé de regarder aussi le clavier logiciel.

Le focus d'attention est un facteur important lors de la mise en place d'une expérimentation : plus le nombre de focus est faible, plus les sujets pourront se concentrer sur leur saisie. Les performances de saisie seront alors plus révélatrices de ce que l'on peut attendre du système si on réalise une expérimentation où il n'y a qu'un seul focus et qu'il est porté sur le système à évaluer. La manière de présenter le texte source à recopier a par conséquent son importance.

2.1.1.2 Que faire recopier ?

Le texte source à recopier peut être de plusieurs types :

- **Un ensemble de mots isolés :** le plus simple consiste à faire saisir des mots isolés comme dans [Evreinova 04], [Magnien 04], [Zhai 03b]. Les utilisateurs peuvent ainsi mémoriser le mot avant de le saisir. Ceci représente l'avantage de ne pas imposer de focus d'attention supplémentaire lors de la saisie du mot. Par contre, la saisie se trouve saccadée car les sujets ne peuvent pas anticiper en fin de mot le début du suivant ;
- **Des phrases courtes :** une alternative consiste à faire saisir des phrases courtes aux sujets [Zhai 01], [Venolia 01] pour qu'ils puissent les mémoriser avant de commencer à les saisir. La saisie de texte est alors plus fluide. Mackenzie et Soukoreff ont proposé un ensemble de phrases équilibrées [MacKenzie 03] à utiliser pour l'évaluation des systèmes de saisie de texte. Ces phrases ont l'avantage d'être représentatives des fréquences d'apparition des caractères et des co-occurences utilisées pour l'anglais. Depuis, plusieurs auteurs se sont basés sur ces phrases pour faire leur évaluation et ont adopté la tâche de recopie de petites phrases [Isokoski 04c], [Pavlovych 04] ;
- **Un texte entier :** cette méthode est plus lourde à mettre en place mais elle est utilisée dans certains cas [Matias 96, Ward 00, Ingmarsson 04]. Elle permet notamment de voir l'évolution des performances sur des saisies de texte assez longues. Par contre,

elle a comme inconvénient principal de demander une attention supplémentaire aux sujets qui ne peuvent pas apprendre le texte à l'avance.

Sur ce dernier type de saisie, un focus d'attention est ajouté car les sujets doivent prendre connaissance du texte à saisir tout en continuant la saisie. Il est difficile de séparer lors de cette tâche, les phases de saisie de celle de lecture du texte. Il serait important dans ce type de situation d'étudier l'activité occulaire dans la tâche de saisie.

2.1.1.3 Comment présenter le texte source à recopier ?

Généralement, le texte source est affiché à l'écran. Dans le cas des mots isolés et des petites phrases courtes, le texte source est affiché et le *feedback* visuel de ce qui est saisi par le sujet est affiché juste en dessous [Magnien 04], [Pavlovych 05]. Dans ce cas de présentation, il n'y a donc qu'un focus d'attention en supplément de celui porté sur le système de saisie de texte testé : l'utilisateur voit d'un même regard ce qu'il doit saisir, ce qu'il a saisi et si il a fait une erreur.

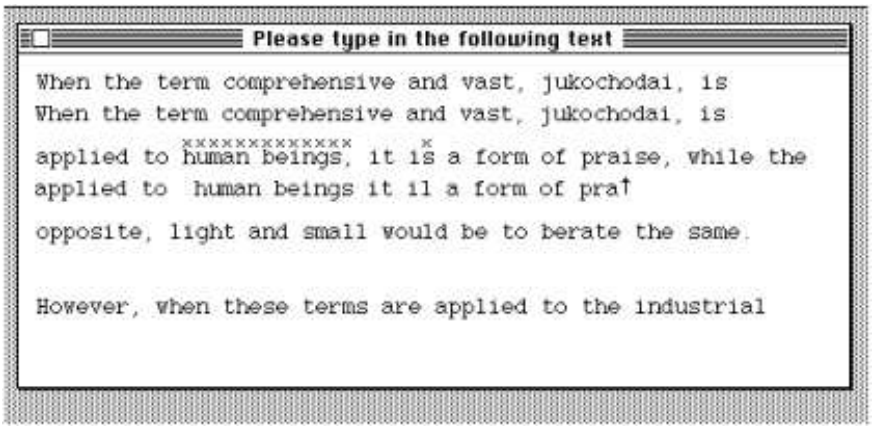


FIG. 6 – Interface de présentation du texte à recopier (repris de [Matias 96])

Dans le cas de la recopie de textes longs, si le texte source est aussi placé intégralement au dessus du *feedback* visuel de ce qui a été saisi, cela crée un double focus d'attention car, par un même regard, le sujet ne peut plus regarder la source et le *feedback* à cause de la longueur du texte. Pour pallier ce problème, il est possible d'intercaler ce que saisit l'utilisateur entre chaque ligne du texte source comme proposé par Matias et ses collègues [Matias 93, Matias 96] (cf. Figure 6). D'autres solutions ont aussi été testées comme, par exemple, n'afficher que les 4 ou 5 prochains caractères à saisir et supprimer au fur et à mesure le caractère qui vient juste d'être saisi pour toujours laisser à la même position sur l'écran le caractère à saisir [Vigouroux 04], [Ingmarsson 04]. Cette technique

a l'avantage de limiter la recherche visuelle à l'écran : le sujet sait où regarder exactement pour connaître ce qu'il a à saisir.

Une solution qui permet de réduire encore plus ce problème de focus d'attention est de dicter au sujet ce qu'il doit saisir. Ward et ses collègues [Ward 00] ont essayé ce mode de présentation. Cependant cette expérience n'a été que peu appréciée par les sujets qui l'ont trouvée plus stressante. De plus, il est plus difficile de savoir si les erreurs de saisie sont dues à un problème d'interaction avec le système de saisie ou si il s'agit d'une mauvaise compréhension de la dictée ou de fautes d'orthographe.

2.1.2 La tâche de création

Cette tâche présente comme principal avantage d'être plus proche de l'utilisation classique du système de saisie de texte. Cependant, peu d'études [Wester 03] utilisent ce type de tâche. [Wester 03] l'a mis en pratique car il ne cherche pas à connaître les performances du système de saisie, mais celles du système de prédiction.

Par contre, dans le cadre de l'évaluation des performances d'un système de saisie de texte, il y a plusieurs désavantages à utiliser cette méthode [MacKenzie 02d] :

- La première d'entre elles, est le fait que l'on ne contrôle plus le comportement de l'utilisateur et les hésitations qu'il pourrait avoir comme par exemple ne pas savoir quoi écrire, avoir peur de faire des fautes d'orthographe. Il devient difficile de savoir si la vitesse de saisie est optimale ou si il y a eu des hésitations qui ont eu pour effet de ralentir sa saisie ;
- Il devient aussi difficile de contrôler les erreurs. On ne peut plus savoir si les erreurs sont dues au système de saisie de texte ou si ce sont des erreurs du type orthographe, conjugaison ou grammaire faites directement par le sujet ;
- A la différence d'une tâche de recopie, le texte qui est saisi librement n'est peut être pas aussi représentatif des fréquences d'apparition des caractères dans la langue en question. Il est par conséquent plus difficile de généraliser les résultats d'une tâche de création que ceux d'une tâche de recopie où le texte avait été étudié de manière à représenter les fréquences d'apparition des caractères dans la langue.

2.2 Les variables à analyser

Pour évaluer les performances d'un système de saisie de texte, deux variables sont analysées : la vitesse de saisie et le taux d'erreur de saisie. Dans les systèmes plus complexes

faisant intervenir des systèmes de prédiction de caractères ou de mots, les performances de ces derniers sont aussi évaluées.

2.2.1 La vitesse de saisie de texte

Il existe deux unités de mesures pour quantifier cette vitesse :

- **Le nombre de caractères saisis par seconde :** en anglais *characters per second (cps)*. Cette mesure s'obtient en comptant le nombre de caractères saisis lors de la tâche demandée et le temps en secondes nécessaire pour réaliser cette dernière. La vitesse est alors obtenue en divisant le nombre de caractères par le temps.
- **Le nombre de mots saisis par minute :** en anglais *words per minute (wpm)*. Il est obtenu en multipliant par 60 le nombre de caractères par seconde et en le divisant par 5. Le nombre 5 vient d'une étude réalisée par Gentner [Gentner 83] qui a déterminé que 5 représente le nombre moyen de caractères par mot[11] dans la littérature anglaise.

2.2.2 Les différentes méthodes de calcul du taux d'erreur

Selon la tâche à réaliser lors de l'expérimentation, il est plus ou moins simple de calculer le taux d'erreur lors d'une saisie. Le plus simple pour calculer un taux d'erreur est de se trouver dans un exercice de recopie de mots (ou de texte).

2.2.2.1 La distance d'édition

La distance d'édition[12] qui sépare une chaîne de caractères A et une chaîne de caractères B est définie par le coût de la suite de transformations élémentaires la moins coûteuse pour passer de A à B.

Il existe trois types d'opérateurs :

- **L'insertion :** Cette opération consiste à ajouter un caractère supplémentaire dans la chaîne B par rapport à la chaîne A. Par exemple : $SOIT \rightarrow SOKIT$;
- **La suppression :** Elle est obtenue lorsqu'un caractère de la chaîne A n'apparaît plus dans la chaîne B. Par exemple : $SORTIR \rightarrow SOTIR$;
- **Le remplacement :** Il consiste à mettre un caractère à la place d'un autre dans la chaîne B. Par exemple : $SORTIR \rightarrow SORTOR$;

[11] Cette étude a été réalisée sans se soucier de savoir si les caractères sont des lettres, des espaces ou de la ponctuation.

[12] La distance d'édition est aussi connue sous le nom de distance de Levenshtein [Levenshtein 66], du nom de son inventeur.

Cette distance est utilisée dans l'évaluation de systèmes de saisie de texte lorsque la tâche demandée aux sujets est de recopier le stimuli (mots, textes, etc.) présenté le plus rapidement possible. Les consignes sont : a) faire le moins d'erreur possible ; b) ne pas se soucier de les corriger et de coller parfaitement au texte à recopier. Dans ce cas, on peut calculer la distance d'édition qui sépare le texte source du texte écrit par le sujet.

Exemple 1 *Voici un exemple de calcul de la distance d'édition repris de [Soukoreff 01]. Prenons comme chaîne de caractères initiale (**I**) '**abcd**', et imaginons la chaîne produite (**P**) par le sujet '**acbd**'. Il existe 3 manières de transformer de manière minimale la chaîne de caractères **I** en **P** :*

- *Supprimer le caractère 'c' et insérer un caractère 'c' après le caractère 'b' ;*
- *Insérer un caractère 'b' après le caractère 'a' puis supprimer le second caractère 'b' ;*
- *Remplacer le caractère 'c' par un caractère 'b', puis remplacer le second caractère 'b' par un caractère 'c' ;*

*Chacune des manières présentées nécessite deux opérations pour passer de **I** à **P**. Par conséquent la distance d'édition est de 2.*

L'algorithme permettant de réaliser ce calcul est présenté dans [Soukoreff 01], lui-même repris de [Kruskal 83].

Cette méthode de calcul du nombre d'erreurs ne peut cependant pas être utilisée seule. Le nombre d'erreurs n'a une véritable importance que par rapport au nombre de caractères correctement saisis. C'est pourquoi, on calcule généralement un taux pour discuter des erreurs faites lors d'une tâche de saisie de texte.

2.2.2.2 Méthodes de calcul du taux d'erreur

Soukoreff et MacKenzie ont proposé en 2001 [Soukoreff 01] un calcul du taux d'erreur fondé sur la distance d'édition. Il propose de calculer le taux d'erreur comme suit :

$$Tx = \frac{MSD(I,P)}{max(|I|,|P|)} \times 100 \tag{16}$$

où $MSD(I,P)$ représente la distance d'édition entre la chaîne de caractères initiale **I** et celle produite par le sujet **P**, qu'ils divisent par la plus grande des deux chaînes de caractères.

Exemple 2

texte source : *bonjour tout le monde*
texte produit : *bon**kjour tou** le mon**f**e*

L'utilisateur n'a fait réellement que 3 erreurs en recopiant le texte : il a inséré le caractère 'k' avant le 'j', puis il a oublié le caractère 't' et enfin a remplacé le caractère 'd' par un caractère 'f'. La distance d'édition est de 3. Le nombre de caractères sur chaque chaîne est de 21. Le taux d'erreur de saisie est donc de 14,29%.

La façon de calculer ce taux d'erreur et les types d'erreurs produites ont été revues par [MacKenzie 02c]. Généralement il existe plusieurs possibilités pour transformer de manière optimale une chaîne de caractères **A** en une chaîne de caractères **B**. C'est pourquoi, leur nouvelle version donne un poids à chaque opérateur et pour chaque lettre la possibilité qu'une erreur se soit produite sur ce caractère à cause de cet opérateur. L'algorithme et des exemples sont donnés dans [MacKenzie 02c].

A partir de ces nouvelles statistiques sur les erreurs, ils ont défini la taille moyenne des alignements. Un alignement est l'ensemble des transformations nécessaires pour passer de la chaîne de caractères **A** à la chaîne de caractères **B**. La taille d'un alignement est obtenue en additionnant le nombre de caractères qu'il faut normalement saisir et le nombre d'insertions nécessaires pour obtenir la nouvelle chaîne de caractères[13]. La taille moyenne des alignements est calculée en faisant la moyenne des tailles des alignements qui donnent une distance d'édition minimum. La nouvelle mesure du taux d'erreur est alors :

$$Tx = \frac{MSD(I,P)}{\bar{S_A}} \times 100 \tag{17}$$

où $\bar{S_A}$ représente la taille moyenne des alignements.

Cette nouvelle mesure du taux d'erreur pourrait être rediscutée. En effet, quel que soit le nombre d'erreurs produites et la taille de la nouvelle chaîne de caractères, ces erreurs ont été faites en essayant de recopier la chaîne de caractères source. Il serait normal de prendre en compte dans la mesure du taux d'erreur, la taille de la chaîne de caractères initiale et non pas la taille maximum entre les 2 chaînes ou bien même la taille moyenne des alignements.

2.2.2.3 Nombre de frappes par caractère

Cependant, il n'est pas toujours possible de calculer un taux d'erreur sur une tâche de recopie. En effet, l'exercice consiste quelques fois à recopier précisement le texte source.

[13]Les opérations de suppression et de transformation ne sont pas pris en compte dans la taille

Selon les claviers évalués, le sujet doit alors corriger ses erreurs. Soit il utilise les touches '*retour arrière*', '*suppr*' etc. ; soit le système n'accepte que les caractères correctement saisis et en cas d'erreurs le sujet n'a qu'à recopier à nouveau le caractère jusqu'à ce qu'il soit correct.

Dans ce cas, il est possible de calculer le nombre de frappes par caractère -Keystrokes per character (KSPC)[MacKenzie 02a] : ceci consiste à comptabiliser le nombre de touches qui ont été réellement frappées pour obtenir exactement le même texte que la source. Les erreurs de frappe sont ainsi prises en compte dans les touches frappées ainsi que les touches qui ont été utilisées pour supprimer ces erreurs ('*retour arrière*', '*suppr*' etc.).

Exemple 3

texte source :	*bonjour tout le monde*
touches frappées :	*bon**k**←**j**our tout le monde*
texte produit :	*bonjour tout le monde*

Dans cet exemple, le texte produit est bien équivalent au texte source, mais 2 touches ont été frappées en plus pour obtenir le même résultat.

La mesure KSPC est obtenue par :

$$KSPC = \frac{Nb_F}{|T|}$$

avec Nb_F qui représente le nombre de touches réellement frappées et $|T|$ le nombre de caractères de la chaîne de caractères T.

2.2.2.4 Comment coupler MSD et KSPC ?

Les deux mesures de l'erreur (MSD et KSPC) telles qu'elles ont été définies ne peuvent pas être utilisées ensemble. En effet, si la consigne est de ne pas se préoccuper de corriger les erreurs, alors le taux d'erreur MSD sera une valeur positive s'il y a eu des erreurs ; par contre la valeur KSPC sera de 1. Inversement, si le sujet doit corriger les erreurs au fur et à mesure de la saisie pour être synchronisé avec la chaîne de caractères source, alors le taux d'erreur sera de 0, et la valeur KSPC sera supérieure à 1 s'il y a eu des fautes corrigées lors de la saisie.

Afin que ces mesures puissent être utilisées conjointement, Soukoreff et MacKenzie dans [Soukoreff 03b] ont distingué différents types de touches frappées selon certains critères :

- ***Correct* (C) :** qui regroupe l'ensemble des caractères correctement saisis ;

- ***Incorrect and Not Fixed* (INF) :** qui constitue l'ensemble des caractères erronés qui apparaissent dans la chaîne de caractères finale produite par le sujet ;
- ***Incorrect but Fixed* (IF) :** qui représente les caractères qui sont des erreurs mais qui ont été corrigés et qui donc n'apparaissent pas dans la chaîne de caractères finale ;
- ***Fixed* (F) :** qui sont les touches qui ont été frappées pour corriger un caractère erroné.

Ils ont alors défini les nouvelles mesures comme étant :

$$MSD = \frac{INF}{C + INF} \times 100 \tag{18}$$

$$KSPC = \frac{C + INF + IF + F}{C + INF} \tag{19}$$

2.3 Bilan

A partir de cette revue des méthodologies et métriques existantes pour l'évaluation de la saisie de texte, une expérimentation "type" semble se dégager ces dernières années. Celle-ci consiste à faire recopier un ensemble de mots ou de petites phrases à un ensemble de sujets. L'analyse des résultats consiste généralement à extraire des données la vitesse de saisie des utilisateurs avec le système en question, ainsi que le taux d'erreurs produites lors de la tâche de saisie.

Néanmoins, cette standardisation des expérimentations laisse encore quelques problèmes en suspens. Par exemple, même si Mackenzie et Soukoreff [Soukoreff 04] mettent à disposition de la communauté leur outil d'analyse des erreurs, aucun outil n'existe encore pour permettre des expérimentations de systèmes de saisie sur une base commune d'évaluation, où il serait plus facile de comparer les différents systèmes.

2.4 Nos choix pour la suite

Nous avons choisi pour nos évaluations de faire de la recopie de mots. Nous avons fait ce choix, car il n'existe pas pour le moment un corpus de petites phrases courtes, comme c'est le cas en anglais [MacKenzie 03], qui soient représentatives de la langue française.

Nous avons donc opté pour un ensemble de mots que nous avons choisi parmi une liste des mots apparaissant les plus fréquemment dans la littérature enfantine[14]. Nous avons

[14]http ://www.eduscol.education.fr/D0102/liste-mots-naturefrequence.txt

en effet considéré que, du fait de leur appartenance à cette liste, les mots à saisir seraient ainsi représentatifs de la langue française standard, et, de plus, ne devraient pas comporter de difficultés majeures pouvant entraîner des réactions d'hésitation lors de l'exécution de la tâche.

Plusieurs critères de sélection ont été pris en compte pour sélectionner un mot plutôt qu'un autre dans cette liste :

- **La longueur :** Chaque mot comprend entre 5 et 8 caractères ;
- **La distance :** La distance a priori parcourue par le pointeur du dispositif de pointage pour saisir ce mot sur un clavier AZERTY.

  Nous avons calculé pour chaque mot cette distance théorique (que nous avons normalisée en la divisant par le nombre de caractères du mot), puis nous avons classé la liste de mots en fonction de cette distance *a priori*.

  De cette liste, nous avons créé 15 sous listes d'une soixantaine de mots chacune. De chacune de ces sous listes, nous avons sélectionné (selon le critère de fréquence décrit ci-dessous) deux mots pour l'ensemble qui servira de corpus de test. Ceci permet d'équilibrer notre corpus avec des mots qui ont des distances très différentes. Ainsi l'évaluation porte à la fois sur des mots dont la proximité des lettres sur le clavier ne pose a priori pas de problème de vitesse, mais aussi sur des mots dont les lettres sont éloignées : ce qui engendre une vitesse de saisie plus faible ;
- **La fréquence :** Dans chaque sous liste, les mots ont été choisis de manière à avoir dans notre corpus de test, l'ensemble des co-occurrences de caractères les plus représentatives de la langue française. Nous avons pris soin de ne pas reprendre trop souvent une même co-occurence de caractère.

Ainsi, nous avons un corpus de test de 30 mots, dans lequel les 100 co-occurences de caractères les plus fréquentes de la langue française sont au moins représentées une fois et au plus cinq.

maladie	voila	nation	question	dernier
ainsi	precis	heureux	fauteuil	pourquoi
rappeler	petit	trois	remplir	arrivee
campagne	sommeil	violent	etrange	dessus
alors	avancer	second	resultat	hasard
bataille	chacun	etouffer	inconnu	assez

TAB. 3 – Liste des mots choisis pour nos expérimentations

Le tableau 3 présente l'ensemble des mots que nous avons retenus pour nos expérimentations futures.

En ce qui concerne les outils utilisés pour effectuer nos expérimentations, nous allons maintenant les détailler au chapitre suivant.

3

La plate-forme E-Assiste

Comme nous avons pu le constater au cours des deux premiers chapitres, il existe une grande diversité de méthodes qui permettent d'évaluer *a priori* ou *a posteriori* un clavier logiciel. Néanmoins, pour qu'une expérimentation soit vraiment intéressante pour un plus grand nombre de personnes, il faut qu'elle respecte quelques propriétés. Celles-ci ont été définies de la manière suivante par MacKenzie dans [MacKenzie 02d] :

> *"An evaluation is valuable and useful if the methodology is* ***reproducible*** *and results are* ***generalizable****.* ***Reproducible*** *implies that other researchers can duplicate the method to confirm or refute results. ...* ***Generalizable*** *implies that results have implications beyond the narrow context of the controlled experiment."*

Nous verrons dans ce chapitre que ces propriétés ne sont que peu respectées ou mal appliquées. Nous en étudierons les causes et proposerons un outil afin d'y remédier : la plate-forme E-Assiste. Cette plate-forme d'expérimentation à la fois générique et personnalisable permet à chaque concepteur de clavier logiciel d'évaluer son système et de le comparer à d'autres sur une métrique commune.

3.1 Problématiques

3.1.1 Des résultats difficiles à comparer

Les expérimentations réalisées lors d'évaluations de systèmes de saisie de texte sont trop souvent le fruit d'un besoin particulier. Le concepteur du système à tester développe souvent lui-même son outil de recueil et d'analyse des données d'évaluation, et compose son propre corpus de tests. Lors de la publication des résultats, il est généralement difficile

de connaître précisément les conditions d'expérimentation. Les auteurs préfèrent souvent mettre en avant les résultats des performances de leurs systèmes plutôt que de décrire le protocole d'évaluation. Si l'on connaît généralement les types d'exercice que les sujets ont dû réaliser, il est plus rare de connaître le texte ou l'ensemble des mots qui devait être recopié, et la procédure exacte de l'expérimentation. Les résultats discutés par les auteurs se résument généralement à des taux d'erreur et des vitesses de saisie (cf. Chapitre 2).

Ce manque de corpus de tests empêche aussi de comparer les performances de systèmes. On peut, par exemple, citer la comparaison (pourtant plus simple) des performances *a priori* de claviers "statiques" qui ne diffèrent que par leur disposition de touches et/ou caractères. En effet, même si la formule 9 de Soukoreff et MacKenzie [Soukoreff 95] fait maintenant office de référence pour ce type d'évaluation, il apparaît des différences significatives dans les performances données dans divers articles. Par exemple, [Zhai 00], [MacKenzie 99a], [Raynal 05] trouvent des résultats différents pour les claviers FITALY et OPTY simplement en modifiant les constantes a et b de cette loi et en calculant la distance séparant deux touches de manière différente.

Ce souci de comparaison est encore plus problématique dès lors que les systèmes de saisie de texte sont augmentés par des comportements dynamiques. Les concepteurs utilisent alors souvent dans leur expérimentation un clavier dit 'de base' pour pouvoir comparer les performances. Ce dernier est généralement le clavier AZERTY ou le clavier QWERTY selon la culture. Mais ce clavier de base (souvent reproduit pour les besoins de l'expérimentation) diffère d'une expérimentation à une autre. Par exemple dans [Venolia 94], [Isokoski 04b], [Vigouroux 04], trois variantes de ce clavier de base sont utilisées. Ces variantes se situent au niveau de la disposition des touches, mais aussi sur les fonctionnalités qui sont proposées sur ce dernier en fonction de celles proposées sur le nouveau système (présence ou non par exemple des caractères 'alt', 'ctrl' ...). A cause de celles-ci il est difficile de comparer (même par interpolation) les différents systèmes entre eux.

Il s'exprime le besoin, comme pour la recherche d'information ou dans le domaine de la reconnaissance automatique de la parole avec les évaluations DARPA, de disposer de corpus de tests et d'évaluations (avec par exemple des exercices de saisie en fonction de classes de systèmes de saisie). Sans ces métriques et ces données, il n'est pas possible de comparer l'évaluation des recherches en inter-système et intra-système des claviers logiciels.

3.1.2 Loin de représenter une utilisation normale

Dans la majorité des cas, les expérimentations ont encore lieu en laboratoire dans des conditions lointaines de celles en usage réel. Les expérimentations sur PDA sont faites généralement assis [Magnien 05, Oriola 05] alors que l'utilisation du PDA est habituellement en mobilité. Les systèmes qui sont souvent faits pour être utilisés sur PDA sont testés sur d'autres dispositifs tels que des tablets PC ou tablets Wacom [Zhai 03b, Isokoski 04b].

Au regard des technologies existantes, il serait intéressant de pouvoir réaliser ces expérimentations *in situ*. Ceci apporterait d'une part, un plus grand confort d'utilisation dans certains cas, et d'autre part des résultats plus représentatifs d'une utilisation quotidienne.

3.1.3 Des évaluations pas assez poussées

Comme relevé dans [Soukoreff 03a], la majorité des évaluations réalisées ne prennent en compte que ce qui est réellement produit par le clavier logiciel et non l'ensemble des interactions qui ont lieu entre le sujet et le système ou entre les divers composants du système. En plus du problème du traitement des erreurs (cf. Chapitre 2), ce traitement de la sortie produite par le système (en général, la chaîne de caractères saisie par le sujet) ne permet pas de prendre en compte certains mécanismes tels que les combinaisons de touches réalisées pour atteindre certaines fonctionnalités du système ou encore les déplacements réalisés avec le pointeur du dispositif de pointage. Il serait intéressant de connaître le comportement du sujet au cours de l'expérimentation en analysant par exemple les déplacements du pointeur, ou ce qui a été proposé par le système de prédiction (quand celui-ci est utilisé) et si son apport a été réellement utilisé et bénéfique au sujet.

Par conséquent, le problème de l'évaluation concerne :

1. le manque de base de tests d'exercices et de consignes qui puissent être utilisées par tous pour réaliser des expérimentations comparables ;

2. la nécessité d'une plate-forme d'évaluation qui regroupe les outils standards (interface de présentation du texte source à recopier par exemple) et des outils d'analyse statistique des métriques ;

3. la possibilité que cette plate-forme fonctionne de manière distribuée pour permettre l'évaluation de dispositifs de saisie en mobilité.

3.2 La plate-forme E-ASSISTE

Notre plate-forme nommée E-ASSISTE [15] a pour vocation de faciliter les expérimentations pour deux types de personnes : a) pour les concepteurs des systèmes à évaluer ; b) pour les sujets qui contribuent à l'évaluation de ces systèmes.

3.2.1 Objectifs

Afin de proposer une solution aux questions soulevées précédemment, nous nous sommes fixé les objectifs suivants pour le bon fonctionnement de notre plate-forme :

- **Standardisation :** Pour faciliter les comparaisons entre les différents systèmes évalués sur la plate-forme, divers standards doivent être définis : par exemple, dans la manière de présenter les informations aux sujets, dans l'analyse des résultats d'expérimentation ou encore dans les exercices de tests utilisés. E-ASSISTE a comme première propriété d'être une base de ressources standardisées pour permettre la comparaison inter et intra systèmes.
- **Généricité :** Pour que les concepteurs puissent facilement ajouter leur système de saisie de texte à notre plate-forme, celle-ci se doit d'être **générique**. Nous entendons par générique, le fait que tout système puisse venir se "plugger" sur la plate-forme et ce quel que soit le langage de programmation et l'environnement qui entoure le système. Cette généricité implique aussi que tous les événements récupérés lors des expérimentations soient génériques.
- **Evaluation *in situ* :** Notre dernier objectif est de faire de cette plate-forme un outil d'évaluation *in situ*. Nous souhaitons que cette plate-forme puisse réaliser tout type d'évaluation de clavier et ce dans n'importe quel contexte d'utilisation : aussi bien en mobilité sur un dispositif portable de type PDA ou encore chez un sujet handicapé qui doit utiliser son propre matériel pour réaliser une expérimentation.

3.2.2 Principe de modularité

Pour répondre à ces objectifs, nous avons conçu la plate-forme E-ASSISTE de manière modulaire. Ce type de conception permet de changer un module par un autre sans modifier le reste de la plate-forme. Un module peut être n'importe quelle partie de ce qui servira à une expérimentation : une interface de présentation de texte pour un exercice de recopie de texte, un système de prédiction, un outils d'analyse des erreurs de saisie, etc. sont des

[15] E-ASSISTE : **E**valuation des **ASSI**stants à la **S**aisie de **TE**xte

exemples de modules indépendants. Un système de saisie de texte est lui même vu comme un module à part entière ou un ensemble de modules de la plate-forme. On peut ainsi réaliser la même expérimentation en changeant simplement le système de saisie de texte à tester.

3.2.3 Architecture

La plate-forme fonctionne à partir de modules répartis en deux parties (cf. Figure 7) :

- **Une partie expérimentation :** C'est la partie de la plate-forme qui est distribuée aux sujets pour effectuer leur expérimentation. Elle comporte tous les modules nécessaires pour qu'un sujet puisse opérer une expérimentation : c'est à dire le système à évaluer (système de saisie de texte + éléments extérieurs tels qu'un système de prédiction par exemple) et le dispositif de présentation de l'exercice à réaliser.
- **Une partie sauvegarde/analyse :** Elle reçoit les traces des différentes expérimentations en cours. C'est sur cette partie que s'effectuent toutes les analyses de traces. Celles-ci peuvent avoir lieu soit sur les traces récupérées en temps réel, soit sur un fichier de sauvegarde des traces d'une ancienne expérimentation.

De plus, pour faciliter la mise en place d'expérimentation, des modules de base comportant les outils standards d'évaluation sont proposés sur la plate-forme E-ASSISTE (voir l'annexe B pour une description complète de ces outils). Il y a par exemple des interfaces qui permettent de réaliser des exercices de création de texte, des tâches de recopie de mots, ou de texte. De la même manière, des modules d'analyse des mesures standard (pour calculer par exemple la vitesse de saisie de texte, le nombre d'erreurs effectuées ou encore la distance parcourue par le curseur du dispositif de pointage) sont proposés sur la plate-forme.

L'ensemble des modules nécessaires pour faire faire une expérimentation à un sujet a été dissocié des modules d'analyse des données recueillies. Cette partie "expérimentation" a l'avantage de pouvoir être séparée du reste de la plate-forme et être distribuée sur différents dispositifs à la fois (cf. Figure 8). Cela permet de pouvoir réaliser plusieurs expérimentations en même temps. La partie analyse se charge de centraliser l'ensemble des données pour pouvoir les analyser par la suite.

3.2.3.1 La communication inter-agents

L'ensemble des messages représentant les interactions qui ont lieu sur la plate-forme ont été standardisés. Comme tous les modules intégrés à la plate-forme utilisent ce standard, ceci permet de pouvoir analyser de la même manière toutes les expérimentations et de les

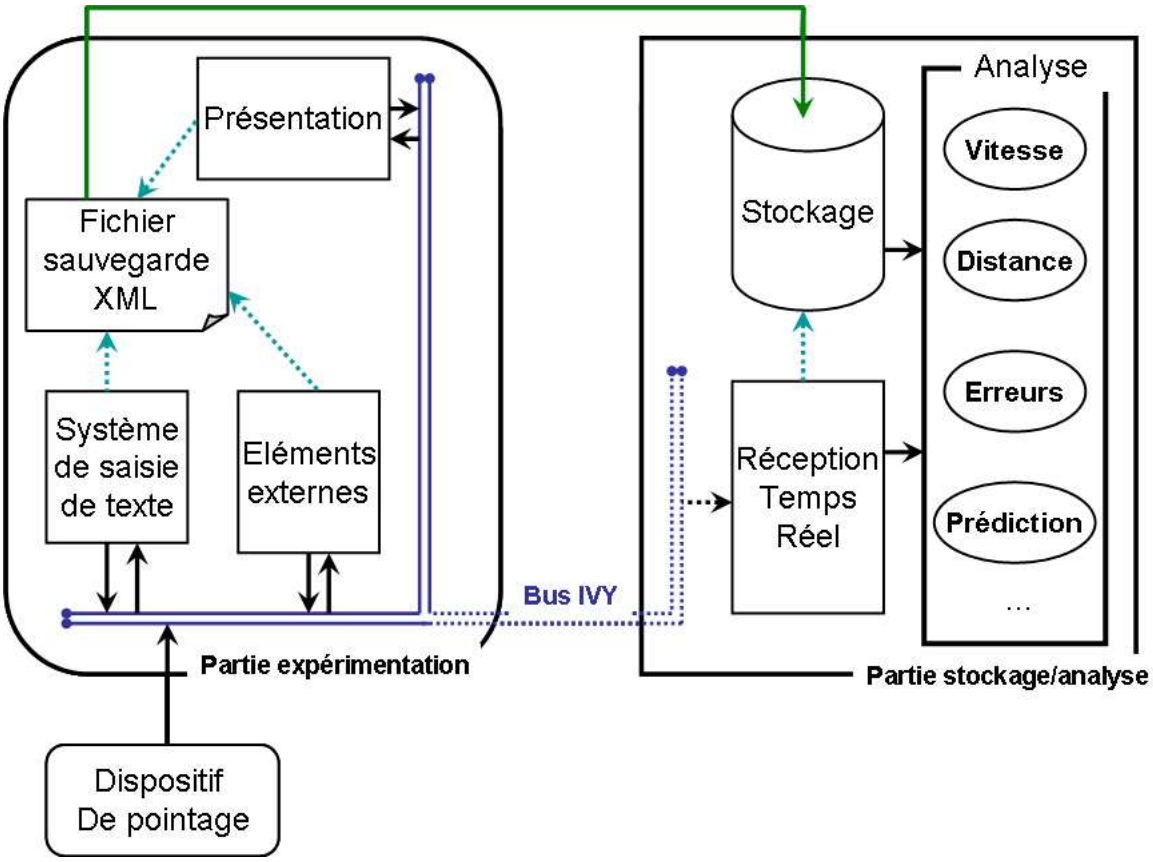


FIG. 7 – Architecture de la plate-forme E-ASSISTE

comparer entre elles. De plus, si cette analyse standard ne suffit pas, chacun peut réaliser une analyse supplémentaire à partir des traces qui sont sauvegardées dans un format générique.

La communication entre les différents modules s'effectue par messages textuels. Pour réaliser la transmission temps réel de ces messages, nous utilisons le bus logiciel IVY [Buisson 02][16]. Ce bus a été réalisé à l'origine pour connecter des composants hautement interactifs. Ainsi tous les modules de la plate-forme sont connectés sur le même bus IVY (même adresse de connexion et même port). Tous les événements (c'est-à-dire l'ensemble des interactions entre le sujet et le système) qui se produisent sur les modules de la partie expérimentation sont envoyés sur le bus IVY . Pour que ces messages puissent être traités

[16]http ://www.tls.cena.fr/products/ivy/.
Celui-ci a l'avantage de pouvoir être utilisé par un grand nombre de langages de programmation et sous tous les environnements. Le principe d'IVY est basé sur l'envoi et la réception de messages textuels. Un module utilisant IVY se connecte à une adresse IP ainsi qu'à un port. A partir de là, il peut envoyer des messages textuels sur cette adresse. Pour la réception de message, un module doit s'abonner à la réception de certains messages. Pour définir ces messages, il peut utiliser les expressions régulières pour récupérer un ensemble de messages ayant la même structure.

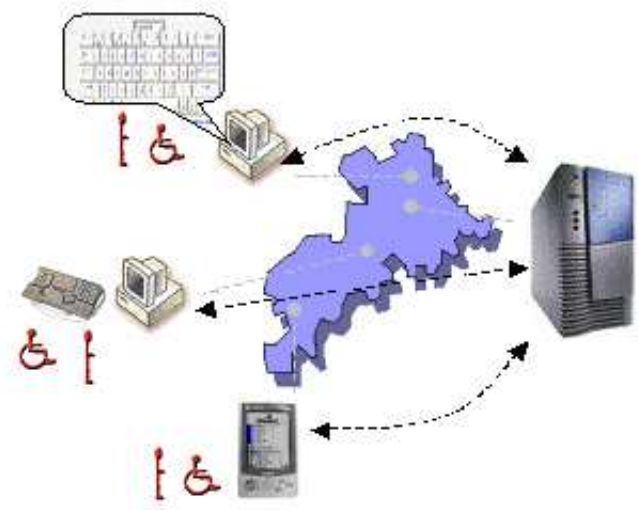


FIG. 8 – E-ASSISTE : une plate-forme distribuée

par un module, chacun d'eux est 'abonné' aux types de message qu'il souhaite recevoir. C'est pourquoi nous avons mis au point un protocole de communication afin d'uniformiser les messages qui transitent sur ce bus et ainsi faciliter leur réception par les autres modules.

Exemple 4 *Illustrons le principe au moyen d'un exemple d'exercice de recopie de mots. Le module de présentation de l'exercice est un module d'affichage du mot à recopier avec le feedback de l'utilisateur en dessous. Les caractères saisis par l'utilisateur ne sont affichés que lorsque ce n'est pas une erreur. Le module de présentation est abonné de manière à recevoir les messages contenant les caractères saisis via le système de saisie de texte. L'utilisateur saisit un caractère sur le système de saisie de texte. Ce caractère est envoyé sous la forme d'un message sur le bus logiciel* IVY *. Le module de présentation le récupère et peut ainsi le traiter pour savoir s'il doit l'afficher ou non.*

De même, un module supplémentaire peut aussi recevoir un message via le bus IVY , le traiter et renvoyer à son tour un ou plusieurs messages sur le bus.

Exemple 5 *Un système de prédiction peut, par exemple, être abonné pour recevoir les messages contenant les caractères saisis sur le système de saisie de texte. Lorsqu'un caractère est saisi par l'utilisateur, ce caractère est encapsulé dans un message et envoyé sur le bus* IVY. *Le système de prédiction le récupère, et renvoie sur le bus le résultat de la prédiction sous la forme d'un nouveau message.*

D'autre part, la partie analyse n'a qu'un module connecté sur le bus IVY. Ce module est chargé de récupérer tous les messages qui transitent sur ce bus. Les autres modules

d'analyse prennent ensuite sur ce dernier les messages qui les intéressent pour faire leur analyse.

3.2.3.2 Protocole de communication

Ce protocole de communication a été mis en place afin de définir tous les types d'événements qui se produisent lors des expérimentations de claviers logiciels. Ce protocole devait être le plus complet possible pour faire apparaître tous les événements qui interviennent dans les différents modules, que ce soit un événement produit suite à une action de l'utilisateur ou suite à un événement interne d'un autre module. On distingue, par exemple, dans ce protocole les messages décrivant les événements provoqués par l'utilisation d'un dispositif de pointage (déplacement de celui-ci, pression ou relâchement d'un de ses boutons), les caractères envoyés par le module de prédiction, etc.

Ces messages sont donc utilisés pour la communication intra-modules, mais aussi lors de l'analyse de l'expérimentation. A partir de l'analyse du contenu de ceux-ci, nous déduisons les actes interactionnels de l'utilisateur.

3.2.3.3 Sauvegarde des événements

En parallèle de leur transmission via le bus logiciel Ivy, les messages sont sauvegardés dans un fichier sur la partie expérimentation. Les messages y sont enregistrés dans un langage balisé qui respecte les règles de la norme XML. Ce langage se nomme KTML (Keyboards Traces Markup Language). Les balises disponibles sont décrites dans l'annexe A.

A la fin de chaque expérimentation, si la partie expérimentation n'est pas connectée via le bus Ivy au reste de la plate-forme E-Assiste, le fichier de sauvegarde est alors envoyé à la partie analyse. Dans le cas où les deux parties sont connectées au bus Ivy, ce fichier de sauvegarde est utilisé pour vérifier si tous les messages ont bien été transmis à la partie analyse : dans le cas où le fichier de sauvegarde est plus grand que celui de la partie analyse, le fichier KTML est envoyé à la partie analyse. Ceci évite ainsi les problèmes de transmission réseau.

Conclusion

Nous avons présenté dans cette partie les deux types de procédés d'évaluation :

- L'évaluation prédictive qui permet de donner un intervalle théorique de vitesse de saisie que pourrait avoir un utilisateur avec le clavier logiciel en question. Cependant, cette méthode n'est pas fiable à 100% : d'après une étude de Mackenzie [MacKenzie 01b], l'estimation des performances d'un utilisateur novice est trop élevée par rapport à celles réellement obtenues en situation. Par contre cette méthode est utilisable pour réaliser des comparaisons entre des claviers dont les dispositions de touches et/ou caractères sont différentes. Ainsi, il est possible de savoir si une disposition est meilleure qu'une autre. C'est pourquoi nous utiliserons cette méthode pour comparer la première optimisation que nous proposons avec les autres méthodes existantes.
- L'expérimentation avec des sujets : cette méthode est plus lourde à mettre en place et plus coûteuse en temps. Cependant elle permet d'avoir une idée un peu plus précise sur le comportement de l'utilisateur et du système, surtout quand ce dernier est dynamique et peut donc engendrer des réactions différentes selon les utilisateurs. Cette méthode est donc conseillée dès lors que l'on veut réellement connaître son utilisabilité par des personnes.

Nous avons aussi présenté un ensemble de métriques qui sont maintenant reconnues comme standard pour l'évaluation de système de saisie de texte. Cependant, devant le manque d'outils et de ressources nécessaires pour conduire à bien une expérimentation, nous avons construit une plate-forme d'évaluation conçue de manière générique afin de pouvoir comparer différents systèmes et ainsi avoir un référentiel commun. En plus de son utilisation dans différents projets (entre autres, le projet CHATCOM[17]), nous avons utilisé cette dernière pour évaluer le clavier augmenté que nous présenterons en dernière partie de ce manuscrit.

[17] http ://www.irit.fr/ChatCom
Projet issu de l'Appel d'Offre Usage et Internet : http ://www.recherche.gouv.fr/appel/2003/resusagesinternet.htm

Perspectives

Face aux premiers besoins d'évaluation de système de saisie de texte, nous avons axé le développement de notre plate-forme sur des modules utiles et utilisables pour des expérimentations sur ordinateur classique. Il nous reste maintenant à développer de nouveaux modules (ou un équivalent de ceux sur ordinateur) pour permettre l'élaboration d'expérimentations sur dispositif mobile à partir de notre plate-forme.

De plus, suite à leurs travaux récents sur le traitement des erreurs lors d'expérimentation de système de saisie de texte [Soukoreff 04], Soukoreff et MacKenzie [18] ont mis à disposition de la communauté scientifique leur outil d'analyse. Nous l'intégrerons comme module de notre application prochainement.

[18] http ://dynamicnetservices.com/w̃ill/academic/textinput

Deuxième partie

Optimisation de la disposition des caractères

Introduction

Comme mentionné en introduction, l'agencement des caractères proposé sur les claviers AZERTY et QWERTY a été conçu de manière à ralentir la vitesse de saisie de texte. Si ceci ne pose pas trop de problème pour de la saisie sur un clavier physique - du fait de l'interaction à 10 doigts - cette disposition des touches et caractères est plus problématique sur un clavier logiciel où la saisie est limitée au seul pointeur du dispositif de pointage. Sur ce type de clavier et avec ce mode d'interaction, l'utilisateur est obligé d'effectuer le déplacement entre chaque caractère à saisir.

Partant de ce constat, de nombreuses recherches ont été menées pour arranger les caractères différemment sur le clavier : d'une part pour minimiser les déplacements (vecteur important de fatigue lors de saisies longues chez les personnes handicapées des membres supérieurs [Bérard 04a, Vella 05]) ; d'autre part, les touches ayant toutes (ou presque) une taille identique sur un clavier logiciel, cette minimisation des distances peut se traduire grâce à la loi de Soukoreff en maximisation de la vitesse de saisie. Nous commencerons cette partie par une étude sur les techniques qui ont été testées pour augmenter la vitesse de saisie de texte.

Nous présenterons ensuite la méthode que nous avons utilisée pour générer à notre tour des agencements de caractères optimisés : une application des algorithmes génétiques au problème de l'agencement des caractères. Après avoir présenté les principes de la méthode et son application (Chapitre 5), nous testerons cette méta heuristique sur la génération de dispositions de caractères pour l'anglais et nous comparerons ses résultats avec ceux des meilleures techniques réalisées jusque là (Chapitre 6, sections 6.1 et 6.1.2).

La plupart des techniques que nous allons présenter ont été proposées et testées pour suppléer le clavier QWERTY lors de saisies de textes en anglais. Or chaque langue a ses spécificités : des co-occurrences qui sont fréquemment utilisées dans une langue ne le seront pas forcément dans une autre.

A titre d'exemple, le tableau 4 donne les cinq co-occurrences les plus fréquentes respectivement dans la langue anglaise et française. Nous donnons leur taux d'apparition

dans chacune des deux langues, ainsi que la position (entre parenthèses dans le tableau) dans le classement des fréquences d'apparition des co-occurrences de l'autre langue.

Anglais				**Français**			
co-oc.	Angl.	Fr.		co-oc.	Fr.	Angl..	
th	3,52	0,07	(210)	er	2,95	1,22	(4)
he	2,94	0,29	(90)	on	2,11	0,55	(45)
an	1,47	0,88	(28)	ra	2,10	0,26	(103)
er	1,22	2,95	(1)	es	2,06	0,58	(43)
nd	1,13	0,27	(96)	nt	2,02	0,35	(77)

TAB. 4 – Comparaison des taux d'apparition des co-occurences les plus fréquentes dans les langues anglaise et française.

Ces différences de fréquences d'apparition entre les langues laissent à penser qu'une optimisation proposée pour la langue anglaise ne sera pas optimale pour le français, et inversement.

C'est pourquoi nous proposerons à la fin de cette partie de réaliser un clavier grâce à notre méta heuristique dont la disposition des caractères sera optimisée pour une saisie de texte en français.

4

Les méthodes existantes

L'objectif de ce chapitre est de présenter une revue des travaux qui ont été menés sur la minimisation des mouvements effectués lors de la saisie de textes. Ces mouvements sont ceux que l'on réalise pour aller pointer la touche qui contient le caractère que l'on veut saisir. Ils sont effectués soit directement au moyen d'un stylet sur un écran tactile, soit par le biais d'un pointeur que l'on déplace à l'aide d'un dispositif de pointage. Nous présentons donc ici l'ensemble des solutions proposées pour répondre à ce problème quel que soit le domaine d'application (dispositif mobile, handicap, etc.) et le type de clavier (logiciel ou physique).

Nous ne parlons dans ce chapitre que des optimisations qui ont été réalisées au moment de générer la disposition des touches et/ou des caractères, c'est-à-dire au moment de la conception du système. Les claviers qui vous seront présentés ici sont dits 'statiques' : c'est-à-dire que la disposition des touches et caractères proposée reste fixe tout au long de la saisie. Les phénomènes dynamiques tels que le ré-arrangement des caractères, la prédiction de caractères ou de mots, etc. seront traités dans la dernière partie de ce manuscrit.

Nous décomposons ce chapitre par rapport à deux grands axes de recherche : tout d'abord, nous traiterons les méthodes qui consistent à réduire le nombre de touches, puis nous étudierons les diverses manières qui ont été proposées pour positionner les caractères de façon à minimiser la distance à parcourir lors de saisies de textes.

4.1 Les claviers réduits

Intuitivement, la manière qui semble la plus efficace pour réduire l'amplitude des mouvements consiste à réduire le nombre de touches présentes sur le clavier. Plusieurs techniques existent afin d'accéder à l'ensemble des caractères d'un clavier complet à partir

d'un nombre de touches moins important. Cependant, beaucoup de ces claviers réduits n'ont été proposés que comme solution matérielle et non sous la forme de clavier virtuel.

4.1.1 Les solutions matérielles

4.1.1.1 Les claviers ambigus

Les claviers dits 'ambigus' associent à chaque touche un ensemble de caractères plus ou moins grand selon le nombre de touches du clavier. On peut par exemple citer la carte SHK [Sugimoto 96], ou encore le clavier réduit QWERTY [Green 04] qui utilisent cette technique pour réduire le nombre de touches.

Les claviers ambigus les plus connus sont les claviers téléphoniques. Ils sont surtout utilisés sur les téléphones mobiles pour l'écriture de SMS (Short Message Service). Ce type de clavier a été utilisé très tôt pour optimiser les assistants à la communication notamment pour la population sourde et muette [Glaser 81, Johnson 81]. La disposition classique[19] consiste à positionner les caractères, trois par trois (sauf pour les touches 7 et 9 qui contiennent quatre caractères), sur les touches en commençant à la touche 2 et en suivant l'ordre alphabétique (cf Figure 9). Cet agencement des caractères, par ordre alphabétique permet à tout utilisateur novice de pouvoir retrouver facilement la touche sur laquelle il doit taper à partir de sa connaissance de l'alphabet.

FIG. 9 – Clavier téléphonique classique

Ces claviers sont dits ambigus car il n'est pas possible à partir d'une seule pression sur la touche de saisir n'importe quel caractère qui y est associé. Il existe deux types de méthodes pour désambiguïser la saisie réalisée au moyen de ce type de clavier[20] :

[19]Cette disposition répond à un standard international [Grover 98]

[20]Pour un état de l'art plus consistant sur la désambiguïsation, nous laissons le lecteur se référer à [Rau 94]

- **Les méthodes directes,** où l'utilisateur contrôle lui même, par ses frappes sur le clavier, le caractère à saisir. La méthode directe la plus connue est *Multi-Tap*, elle consiste à appuyer une, deux ou trois fois sur la touche pour saisir respectivement le premier, second ou troisième caractère présent sur la touche. Cependant, cette méthode a l'inconvénient de nécessiter un grand nombre de frappes sur les touches pour saisir du texte : [Uguen 05] rapporte qu'il faut en moyenne 2,23 et 2,03 frappes par caractères pour saisir un texte respectivement en français et en anglais.

  Pour pallier ce problème, plusieurs ré-arrangements de caractères ont été proposés : soit en laissant les caractères sur la même touche que le clavier téléphonique classique, mais en les ré-arrangeant en fonction de leur fréquence d'apparition, tels que le clavier *Less-Tap* [Pavlovych 03] ; Soit les caractères ont été repositionnés de manière générale sur le clavier [Levine 87, Arnott 92, Lesher 98, Isokoski 04a].

  Ces réarrangements permettent ainsi de diminuer le nombre de frappes à effectuer sur chaque touche afin de saisir un caractère : la disposition Less-Tap nécessite en moyenne 1,52 frappes par caractère, et les claviers présentés dans [Lesher 98] nécessitent 1,14 frappes par caractère.

  Une autre méthode directe est la méthode *Two-Key* [Sirisena 02]. Celle-ci consiste à saisir chaque caractère au moyen de deux pressions de touches. La première pression est réalisée sur la touche qui contient le caractère voulu, et la seconde détermine la position de ce dernier par un appui sur la touche 1, 2 ou 3 pour sélectionner respectivement le premier, second ou troisième caractère de la première touche qui avait été frappée.

  Exemple 6 *Voici comment le mot "jouet" est saisi sur un clavier téléphone tel que celui de la Figure 9 au moyen des méthodes Multi-Tap et Two-Key :*

  Multi-Tap : 5 666 88 33 8

  Two-Key : 51 63 82 32 81

- **Les méthodes indirectes,** qui consistent à utiliser des connaissances linguistiques pour désambiguïser la saisie de l'utilisateur. La plus connue d'entre elles est la méthode T9[21] (des méthodes similaires à T9 existent : on peut notamment citer iTap[22] ou encore eZiText[23]) : celle-ci consiste à n'appuyer qu'une seule fois sur la touche comportant le caractère à saisir. A chaque frappe de touche, le système propose le caractère qui lui paraît le plus pertinent pour succéder au debut de la

[21] Tegic communication Inc. : http ://www.tegic.com
[22] http ://www.motorola.com/lexicus/html/itap.html
[23] http ://www.zicorp.com

saisie. Dans le cas où plusieurs mots pourraient correspondre à la suite de touches frappées, le système propose en premier choix le mot le plus fréquent. L'utilisateur peut modifier ce choix par les touches directionnelles. La fin du mot est marquée par un appui sur la touche 0. Ce système a donc l'avantage de nécessiter moins de frappes pour saisir un mot. En reprenant l'exemple du mot *jouet*, un utilisateur ne frappe que 6 touches avec le système T9 contre 9 et 10 avec respectivement *Multi-Tap* et *Two-Key*. Une étude de [Silfverberg 00] rapporte une vitesse de saisie de l'ordre de 41 à 46 mots par minute pour un utilisateur expert et pour une saisie où la désambiguïsation serait bonne (c'est-à-dire que chaque mot saisi appartient au corpus du T9). Cependant, cette méthode a l'inconvénient de se baser sur un corpus de mots pour lever les ambiguités. Si le mot que l'utilisateur souhaite saisir n'appartient pas à ce corpus, il n'a alors aucune chance d'apparaître.

C'est pourquoi, pour résoudre ce type de problème, Mackenzie et ses collègues proposent la méthode LetterWise [MacKenzie 01a] qui consiste à désambiguïser la saisie caractère par caractère. Après chaque caractère saisi, les caractères présents sur chaque touche sont proposés en fonction de leur probabilité d'apparition en fonction du début du mot. Ainsi, si le caractère 'c' a plus de chance d'être utilisé que les caractères 'a' et 'b', une seule frappe sur la touche 2 suffira pour saisir le caractère 'c'. Cette méthode a ainsi l'avantage de diminuer le KSPC (1,15 contre 2,03).

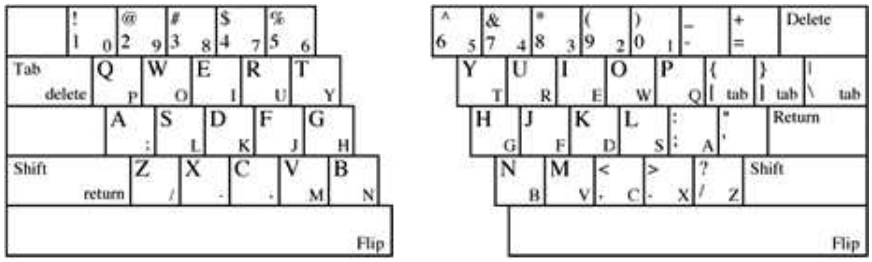
4.1.1.2 Les claviers à accords

Les claviers à accords sont une autre forme de claviers réduits. La saisie de caractères sur ce type de clavier se fait par l'appui simultané de plusieurs touches : un caractère est associé à chaque combinaison. Avec cette technique, il est ainsi possible de coder $2^M - 1$ caractères à partir de M touches. Le clavier Twiddler[24] est un exemple de clavier à accords (cf Figure 10).

Le clavier Half-Qwerty de Matias et ses collègues [Matias 93, Matias 96] (cf Figure 11) est une méthode hybride entre le clavier physique classique et le clavier à accords. Il consiste à saisir du texte sur un clavier classique, mais en n'utilisant que la moitié du clavier. Les caractères de l'autre moitié du clavier sont obtenus par la combinaison de la touche espace et d'une autre touche de la moitié de clavier utilisée. Cette technique permet ainsi une saisie à une seule main. La vitesse de saisie sur ce type de dispositif est de 24 à 43 mots par minute après une dizaine d'heures d'entrainement.

[24]Clavier proposé par la société HandyKey Inc. : http ://www.handykey.com/

FIG. 10 – Clavier Twiddler



A) B)

FIG. 11 – Le clavier Half-QWERTY : partie du clavier pour une utilisation, A) avec la main gauche ; B) avec la main droite

4.1.1.3 Saisie par séquence

Une dernière forme d'interaction consiste à saisir un caractère par une séquence de touches frappées successivement. On peut notamment citer les travaux de Evreinova et ses collègues [Evreinova 04] qui n'utilisent que les quatre flèches directionnelles du clavier physique pour réaliser les combinaisons. Sur ce même principe, il existe aussi les systèmes MDITIM [Isokoski 00] et Glyph [Uguen 05] où chaque caractère est codé par une séquence de primitives. Ces dernières peuvent ensuite être représentées sous plusieurs formes : gestes à effectuer au stylet, boutons d'un clavier physique ou logiciel, etc.

Cependant, dans le cas d'une utilisation sur claviers réduits, la vitesse de saisie de texte n'est pas très haute. Ceci est dû (entre autre) au nombre assez élevé de frappes à réaliser en moyenne par rapport au nombre de caractères saisis. Bien que le KSPC soit

inférieur à la méthode Multi-Tap (1,75 et 1,94 contre 2,23 et 2,03 respectivement pour la langue française et anglaise, d'après [Uguen 05]), celui-ci est plus élevé que celui obtenu par désambiguïsation linguistique par exemple.

4.1.2 Solutions logicielles

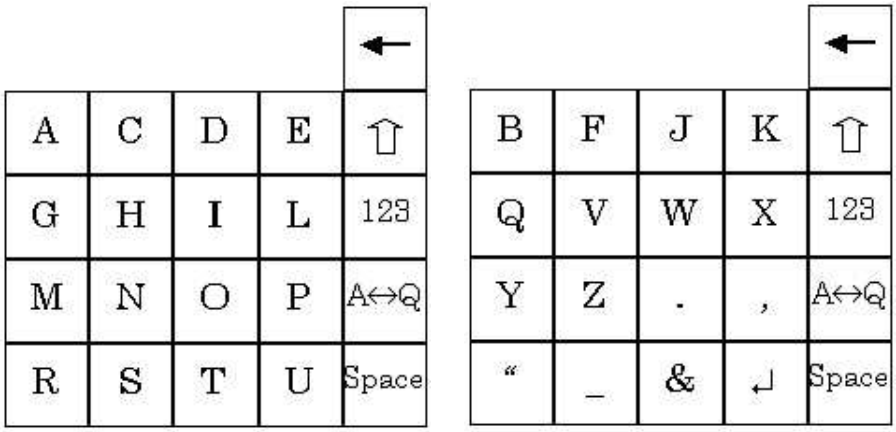


FIG. 12 – Le clavier DotNote : A) Interface contenant les caractères les plus fréquents ; B) Interface avec les caractères les moins utilisés.

Les claviers logiciels réduits sont moins nombreux. Il existe le clavier DotNote[25] qui réduit de moitié le nombre de touches à l'écran (cf. Figure 12). Le clavier présente les caractères sur deux interfaces : l'une contient les caractères les plus fréquemment utilisés et l'autre les moins fréquents. L'utilisateur passe d'une interface à une autre grâce à la touche Shift. Cependant, d'après [MacKenzie 02d], les performances maximales que pourrait avoir *a priori* un expert avec DotNote sont inférieures à celles qu'il pourrait avoir avec un clavier QWERTY.

L'autre solution proposée sous forme logicielle est la disposition téléphonique. Rappelons une étude de Mackenzie et ses collègues [MacKenzie 99b] qui ont testé deux claviers logiciels de type téléphone avec un ordonnancement alphabétique (Figure 13 A.) et Just-Type (Figure 13 B.) initialement proposé par King et ses confrères [King 95] . Les résultats de cette étude montrent que théoriquement il est possible d'avoir de meilleures performances avec un clavier de type téléphone qu'avec un clavier QWERTY. L'augmentation est cependant minime pour des utilisateurs experts : ils estiment à partir du modèle de

[25]Le clavier DotNote est produit par Utilware : http ://utilware.com

Soukoreff (cf. section 1.3.2) qu'un utilisateur expert aurait des performances de 43,5, 44,2 et 43,2 mots par minute respectivement avec le clavier Telephone, JustType et QWERTY.

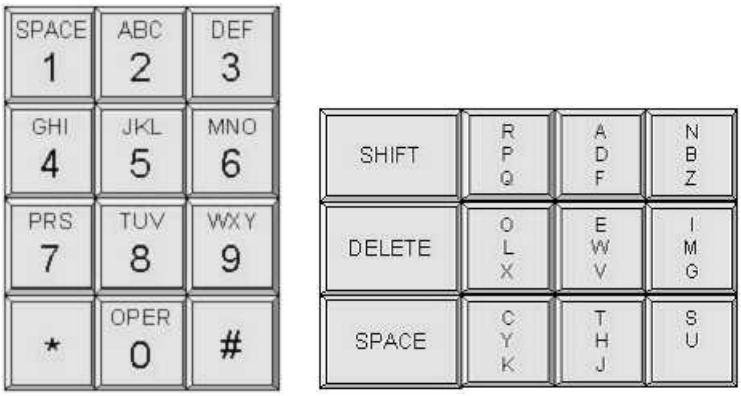


FIG. 13 – Claviers téléphonique testé dans [MacKenzie 99b] : A) Clavier téléphonique classique ; B) Clavier JustType.

Dans le cadre d'une recherche d'assistants à la communication pour "tous", nous avons souhaité étudier l'utilisation de ce type de clavier téléphonique par des personnes handicapées moteur en les comparant à celle d'un clavier AZERTY.

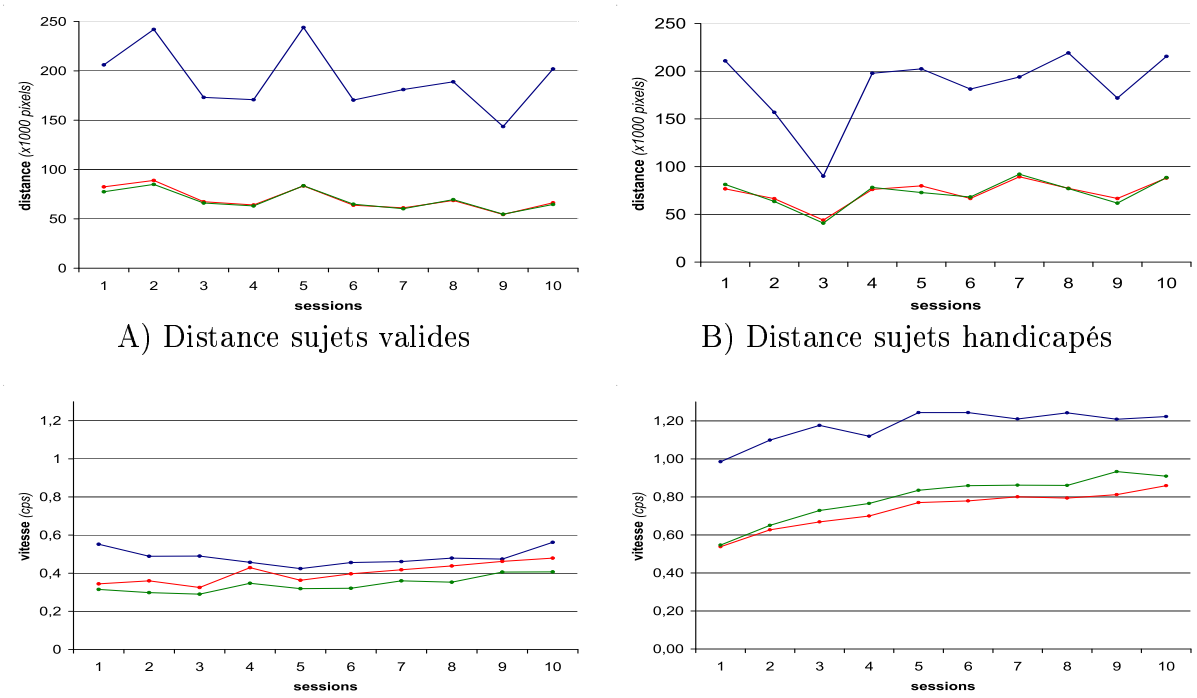
4.2 Analyse de claviers virtuels type téléphone pour un usage par des handicapés moteur

L'expérimentation consistait à tester 3 claviers logiciels : un clavier AZERTY, un clavier type téléphone et le même clavier téléphonique mais en y rajoutant un système de prédiction de caractères. Ce système de prédiction a pour fonction de ré-agencer les caractères sur chaque touche après chaque saisie de l'utilisateur. Les caractères sont positionnés en fonction de leur fréquence d'apparition après celui qui vient d'être saisi. Ainsi sur chaque touche, le caractère qui a le plus de chances d'être saisi est placé en première position (et par conséquent ne nécessite qu'un seul clic sur la touche pour y accéder), le second en seconde position et celui qui a le moins de chance d'apparaître en troisième position. Cette prédiction permet de réduire le nombre de clics à réaliser sur les touches pour pouvoir saisir du texte.

Nous avons formulé l'hypothèse qu'un sujet valide réaliserait plus rapidement les déplacements d'une touche à une autre qu'un handicapé moteur ; ainsi la diminution des distances sur le clavier téléphonique devrait être plus bénéfique en temps pour ces derniers.

Les utilisateurs (valides) experts ayant de meilleurs résultats de vitesse avec le clavier type téléphone qu'avec un clavier QWERTY [MacKenzie 99b], si notre hypothèse est vérifiée, le gain en temps pour des personnes handicapées par rapport au clavier AZERTY devrait être plus important que celui des personnes valides.

Cette expérimentation a été réalisée par deux personnes handicapées des membres supérieurs et cinq personnes valides. Celle-ci avait lieu sur un PC de bureau. Les deux sujets handicapés, du fait de leur manque de motricité, ont utilisé une track-ball, alors que les sujets valides ont utilisé une souris classique. L'expérimentation était découpée en 10 sessions (à raison d'une session par jour). Chaque session consistait à recopier avec chacun des trois claviers un extrait d'environ 200 caractères du texte '*le vieil homme*'[26]. Pour éviter les effets d'apprentissage du texte d'un exercice à un autre, l'ordre d'apparition des claviers a été contre-balancé d'une session à une autre.



A) Distance sujets valides

B) Distance sujets handicapés

C) Vitesse sujets valides

D) Vitesse sujets handicapés

FIG. 14 – Comparaison des résultats AZERTY vs. clavier téléphonique sur 2 populations : valides et handicapés moteur

La figure 14 présente l'évolution au cours des sessions de la distance parcourue avec le pointeur pour saisir le texte (Figure 14 A. et B.), de la vitesse de saisie de texte avec les différents claviers testés (Figure 14 C.et D.). Les distances sont exprimées en nombre de

[26]texte disponible dans [Vigouroux 04]

pixels parcourus à l'écran par le pointeur du dispositif de pointage. La vitesse est donnée en nombre de caractères saisis par seconde en moyenne sur la session. Les courbes bleue, verte et rouge représentent respectivement le clavier AZERTY, le clavier téléphonique classique et le clavier téléphonique avec système de prédiction. Nous pouvons constater que de manière générale que ce soit avec ou sans système de prédiction, le clavier logiciel de type téléphone nécessite moins de déplacements pour saisir du texte que le clavier AZERTY. Cependant, il est important de noter que ce dernier reste plus efficace en terme de vitesse de saisie de texte qu'un clavier type téléphone. Nous remarquons aussi qu'un sujet valide est moins performant avec le clavier téléphone qu'avec le clavier AZERTY. On peut expliquer cette différence par rapport à l'étude de MacKenzie par le fait que le dispositif de pointage est différent entre les deux expériences (stylet vs souris). Cette différence de dispositif entraîne des performances différentes [MacKenzie 91b] à l'avantage du stylet. De plus, cette différence peut aussi s'expliquer par le fait qu'il faut plus de frappes par caractère en français par rapport à l'anglais (2,23 contre 2,03 [Uguen 05]).

L'hypothèse donnée en début d'expérience, à savoir que le clavier téléphone serait plus bénefique aux personnes handicapées qu'aux personnes valides, est vérifiée. En effet, la différence de vitesse de saisie entre les sujets valides et les sujets handicapés est moins importante pour le clavier téléphone que pour le clavier AZERTY . Cependant leur performance avec le clavier téléphone reste moins élevée qu'avec le clavier AZERTY . La différence se réduit encore plus lorsque les sujets handicapés utilisent le clavier téléphonique couplé au système de prédiction. Il semble donc que la succession des clics ne soit pas bénéfique pour la vitesse de saisie de texte des personnes handicapées.

En conclusion de cette expérimentation, bien que les distances à parcourir soient réduites sur le clavier téléphone, ce type de clavier ne paraît pas être une solution envisageable pour des personnes handicapées, à moins que le clavier soit couplé à un système de prédiction. En effet, la répétition de clics successifs pour pouvoir accéder au deuxième ou troisième caractère de la touche est une contrainte motrice pour les personnes handicapées. Il semble par conséquent que cette limite, ajoutée au temps d'appropriation initial, ne favorise pas la prise en main d'un tel dispositif par des personnes handicapées des membres supérieurs. Néanmoins, il serait intéressant de tester d'autres méthodes qui réduiraient encore plus le nombre de frappes à réaliser, et ainsi voir si avec une réduction un peu plus importante, il ne serait pas possible pour les personnes handicapées moteur d'obtenir des vitesses meilleures avec un clavier réduit.

4.3 Méthodes d'agencement des caractères

Comme nous l'avons dit précédemment, une majorité de recherches sur le domaine a lieu dans des pays anglo-saxons, où la saisie de texte ne comporte que peu d'accents. De ce fait, les travaux menés sur la saisie de texte ne sont généralement effectués que sur les 26 lettres de l'alphabet latin et sur le caractère espace. La majorité des claviers que nous allons vous présenter sont réduits à ces 27 touches. Le principe du ré-arrangement des caractères est de rapprocher les caractères qui ont le plus de chance de se succéder afin de limiter les déplacements d'un caractère à un autre lors de la saisie.

4.3.1 Les claviers alphabétiques

C'est la disposition de touches la plus simple voire "naturelle" que l'on puisse faire. Le principe est de mettre les caractères les uns à la suite des autres dans l'ordre alphabétique. Si cet agencement ne permet pas de diminuer les déplacements, il a néanmoins l'avantage d'être connu par tout le monde. Ainsi normalement, une personne novice avec un clavier retrouvera plus facilement les caractères. Norman et Fisher [Norman 82] ont proposé cette disposition alphabétique pour les claviers physiques. Ils l'ont testée pour la langue anglaise. L'expérience ne fut que peu concluante : les sujets novices ne tapèrent pas plus vite sur ce type de clavier que sur un clavier QWERTY. Le principal problème souligné provient de la discontinuité provoquée par le découpage de l'alphabet pour disposer les caractères sur plusieurs lignes. Mackenzie et ses collègues [MacKenzie 99b] proposèrent de réduire ce problème en réalisant un clavier à deux lignes contenant chacune 13 caractères (cf. Figure 15 B.). Mais la distance à parcourir entre les caractères les plus éloignés diminue très fortement la vitesse de saisie que l'on peut avoir avec ce système.

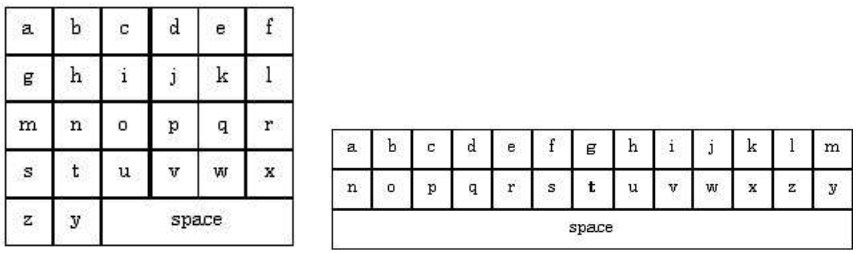


FIG. 15 – Claviers alphabétiques : A) Configuration 5×6 proposée par [Lewis 99b] ; B) Configuration 3×13 proposée par [MacKenzie 99b].

Quant à Soukoreff [Soukoreff 02], il a testé quatre dispositions de touches pour les claviers logiciels alphabétiques. Les résultats *a priori* que l'on pourrait espérer avec

ces claviers alphabétiques ne sont pas beaucoup plus élevés (voire même plus bas pour certaines configurations) que ceux d'un clavier AZERTY. D'un point de vue théorique, ces claviers, comme les claviers AZERTY ou QWERTY, ne permettent pas *a priori* aux utilisateurs experts d'avoir des performances très élévées avec une saisie à un doigt.

4.3.2 Agencement réalisé intuitivement

Le premier clavier conçu de manière intuitive est le clavier physique DVORAK [Dvorak 36]. Le but était d'optimiser la saisie à deux mains sur ce clavier tout en gardant la même disposition des touches que le clavier QWERTY . L'agencement des caractères a été réalisé en rapprochant les caractères qui ont le plus de chance de se succéder dans la langue anglaise de manière à diminuer les mouvements des mains et des doigts sur le clavier. Bien qu'apportant un gain en vitesse de saisie de texte de 4 à 5% [Buzing 03] par rapport au clavier QWERTY, celui-ci n'a pas suffi pour convaincre les utilisateurs de machine à écrire d'abandonner le QWERTY pour DVORAK. Une explication plausible est que le gain apporté est trop faible par rapport au temps d'apprentissage nécessaire pour connaître aussi bien ce clavier que le clavier QWERTY .

Le clavier DVORAK a ensuite été proposé sous forme de clavier logiciel. [MacKenzie 99b] a mesuré les performances qu'aurait un utilisateur avec ce type de clavier dans le cadre d'une saisie au stylet. Cette évaluation a montré qu'un utilisateur du clavier DVORAK n'atteindrait pas les performances de saisie qu'il pourrait avoir avec un clavier QWERTY et ce quel que soit le niveau d'expertise avec le clavier. Ceci s'explique par l'optimisation faite pour une utilisation à deux mains. Des caractères qui se succèdent fréquemment ont pu être placés aux deux extrémités du clavier pour être saisis chacun par un doigt de chaque main. Ils seront ainsi saisis rapidement sur un clavier physique. Par contre, ces caractères seront saisis beaucoup moins rapidement avec un stylet du fait du grand mouvement à réaliser pour parcourir tout le clavier.

Le clavier Chubon [Chubon 88] fut l'un des premiers à être conçu sous forme logicielle et optimisé pour une utilisation à un pointeur. Il a été réalisé dans l'optique d'aider les personnes handicapées à saisir du texte grâce à un seul pointeur. Selon les évaluations *a priori* de [MacKenzie 02d], le clavier Chubon permettrait à un utilisateur expert de gagner plus de 20% en vitesse de saisie par rapport à un clavier QWERTY.

Les claviers logiciels les plus connus pour avoir une disposition de touches/caractères optimisée pour une saisie à un pointeur sont FITALY[27] et OPTI [MacKenzie 99a] (cf. respectivement Figure 16 B. et Figure 16 A.). Ils ont été conçus dans le but de minimiser

[27] commercialisé par Textware Solutions Inc. :http ://www.textwaresolutions.com/

les distances à parcourir dans une tâche de saisie au stylet sur dispositif mobile. Pour cela, les caractères les plus utilisés ont été placés au centre du clavier et les caractères qui ont le plus de chance de se succéder ont été placés à proximité. De plus, plusieurs touches espace (qui est le caractère le plus utilisé) sont positionnées sur ces claviers. Ces claviers ont donc l'avantage d'améliorer les performances des utilisateurs experts. Par contre, l'arrangement des touches n'étant que peu ordonné par rapport à ce que l'on connaît, les performances des utilisateurs novices sont assez basses.

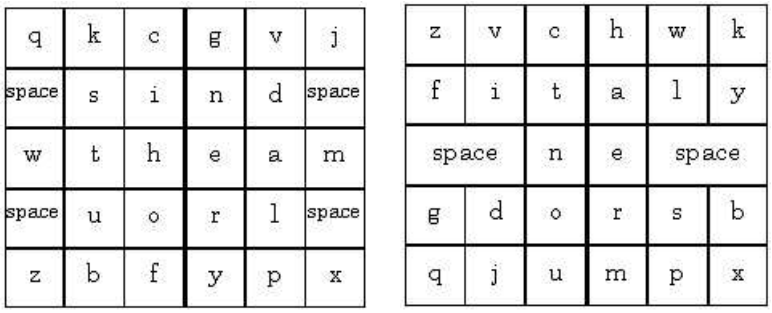


FIG. 16 – Claviers réalisés de manière intuitive : A) Clavier OPTI ; B) Clavier FITALY.

D'après différents tableaux comparatifs [MacKenzie 02d, Zhai 02a] (cf. Tableau 5) ces claviers optimisés de manière manuelle et intuitive présentent des performances *a priori* bien supérieures à celles que l'on peut espérer atteindre avec un clavier logiciel QWERTY : les claviers OPTI et FITALY améliorent d'environ 40% les performances théoriques de vitesse de saisie de texte du clavier QWERTY.

Cependant, sur un clavier, le nombre de combinaisons possibles où les caractères sont positionnés de manière différente est bien trop élevé pour pouvoir être testé dans son ensemble par un humain. En faisant ce type d'optimisation, il est donc possible d'apporter une solution correcte mais certainement pas optimale. C'est pourquoi, des méthodes ont été reprises de problèmes d'optimisation classiques et appliquées à ce problème d'agencement des caractères.

4.3.3 Agencement obtenu avec techniques d'optimisation automatiques

Une autre méthode pour obtenir des dispositions de touches et de caractères optimisées consiste à appliquer un algorithme d'optimisation qui soit en mesure de calculer plusieurs solutions possibles au problème et de n'en garder que les meilleures.

Clavier	Performances théoriques d'un expert (wpm)	Augmentation par rapport à QWERTY (%)	Fig.
Metropolis II	**42,94**	*42,9*	17 B.
OPTI II	**42,37**	*41,0*	16 A.
FITALY	**41,96**	*39,7*	16 B.
Hook's	**41,15**	*37,0*	17 A.
Cubon	**37,02**	*23,2*	-
Lewis	**34,65**	*15,3*	-
ABC III	**32,50**	*8,2*	15 A.
ABC IV	**30,18**	*0,5*	15 B.
Qwerty	**30,04**	-	-
DotNote	**29,46**	*-1,9*	**??**

TAB. 5 – Extrait du tableau de [MacKenzie 02d] donnant les vitesses théoriques pour les claviers présentés dans ce manuscrit

4.3.3.1 Des méthodes de recherches incomplètes

Le premier clavier conçu de la sorte est le clavier Getschow [Getschow 86]. Pour créer ce clavier, Getschow utilisa le '*greedy algorithm*' (ou algorithme par construction). Cet algorithme consiste à rajouter au fur et à mesure des itérations un élément du problème. Une recherche en profondeur est réalisée à chaque itération sur le nouvel ensemble d'éléments. Getschow a appliqué cet algorithme à l'agencement de caractères en divisant le problème en trois éléments. Il commence par prendre les neuf caractères les plus fréquemment utilisés dans la langue anglaise et recherche la disposition de ces neuf caractères qui minimise la distance moyenne entre eux tout en pondérant la distance entre deux caractères par la fréquence d'apparition de cette co-occurence. Une fois la meilleure solution trouvée, il fige cette disposition de caractères, rajoute les huit caractères suivants autour de cette disposition et recommence la recherche de la meilleure solution en testant les différentes combinaisons possibles en positionnant différemment ces huit caractères autour des neuf premiers. Une fois la meilleure solution trouvée, il fige les 17 premiers caractères, puis recherche la meilleure disposition possible autour de ces 17 pour les huit caractères restants. Cette méthode a néanmoins l'inconvénient de ne pas parcourir l'ensemble des solutions. En

effet, une fois les caractères fixés lors d'une itération, il n'est plus possible de modifier leur position en même temps que les caractères de l'itération suivante. Il est par conséquent impossible de tester par cette méthode une disposition de caractères où, par exemple, un ou plusieurs caractère(s) moins fréquemment utilisé(s) serai(en)t au centre du clavier.

Lewis et ses collègues [Lewis 99a, Lewis 99b] proposent eux de remplir une matrice symétrique avec la fréquence d'apparition de chaque co-occurence. A partir de cette matrice, ils cherchent le chemin passant par tous les caractères qui minimise au mieux les distances pour les plus fortes co-occurences. Cette méthode à l'inconvénient de ne pas prendre en compte toutes les connexions possibles qu'il y a sur un clavier : la recherche du meilleur chemin consiste à regarder après chaque caractère sélectionné quel est celui qui a le plus de chance de lui succéder pour le placer à côté de ce dernier. Toutes les autres connexions avec ce caractère sont ignorées : un caractère qui lui succède de façon assez fréquente peut se retrouver à l'autre bout du clavier. Cette méthode donne un résultat correct mais ne certifie en rien que l'on a une des meilleures solutions.

4.3.3.2 Des méthodes de recherche par voisinage

Lesher s'est fixé comme objectif de n'optimiser que l'agencement des caractères. Il proposa le Clavier KNITS [Lesher 00]. Pour cela, il commença par figer la disposition des touches sur lesquelles il associe un caractère. Il utilise ensuite l'algorithme *n-opt*[28] pour rechercher le meilleur agencement possible pour ces caractères. Cette méthode est dite de 'voisinage' : elle consiste à échanger la position de deux caractères sur le clavier et tester si cette nouvelle configuration est meilleure que la précédente. Dans le cas contraire, l'échange n'est pas pris en compte. L'opération est réalisée autant de fois que l'on veut, souvent jusqu'à obtenir une solution acceptable.

Zhai et ses collègues [Hunter 00, Zhai 00] ont proposé à leur tour des claviers optimisés. A la différence des autres travaux décrits ci-dessus, le caractère est associé à une touche. Ainsi, lors de l'optimisation, la touche est déplacée avec le caractère. Par conséquent, on ne connaît pas à l'avance la forme qu'aura le clavier[29]. Ils ont essayé deux techniques inspirées de la physique et de la thermodynamique pour générer deux nouveaux claviers :

- **Le clavier Hooke** [30] (cf. Figure 17 A.) où chaque liaison entre deux caractères est vue comme un ressort. L'élasticité du ressort est proportionnelle à la fréquence d'apparition de la co-occurence. La distance entre deux caractères constitue la

[28] L'algorithme *n-opt* a été initialement proposé par Lin [Lin 65, Lin 73] pour résoudre le problème du voyageur de commerce

[29] Leur méthode a aussi été réalisée de manière à pouvoir figer les touches au départ et ne déplacer par la suite que les caractères sur une disposition de touche donnée

[30] du nom de la loi de Hooke

tension du ressort. Celle-ci est pondérée par l'élasticité du ressort. L'état de départ est un positionnement aléatoire des caractères. La simulation consiste à déplacer les touches de manière à rapprocher les caractères afin d'obtenir un système où la tension de chaque ressort est minimale.

- **Le clavier Metropolis** (cf. Figure 17 B.) du nom de l'algorithme utilisé pour générer ce clavier. L'algorithme Metropolis [Metropolis 53] est à la base utilisé pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température T donnée. A partir d'un système initial, il consiste à modifier légèrement le système par itérations successives : le but est de faire diminuer l'énergie du système. Ainsi à chaque itération, si la modification apporte une diminution de l'énergie, la modification est acceptée. Dans le cas inverse, la modification peut quand même être acceptée avec la probabilité $e^{\frac{-\Delta E}{T}}$. Dans le cas de l'optimisation des claviers la fonction énergie est matérialisée par la loi prédictive de Soukoreff (cf. équation 9). Proposer une modification consiste à déplacer un ou plusieurs caractères.

D'après les résultats comparatifs des performances que l'on pourrait avoir *a priori* avec ces différentes dispositions optimisées, Metropolis ressort comme étant la meilleure [Zhai 02a, MacKenzie 02d] (cf. Tableau 5).

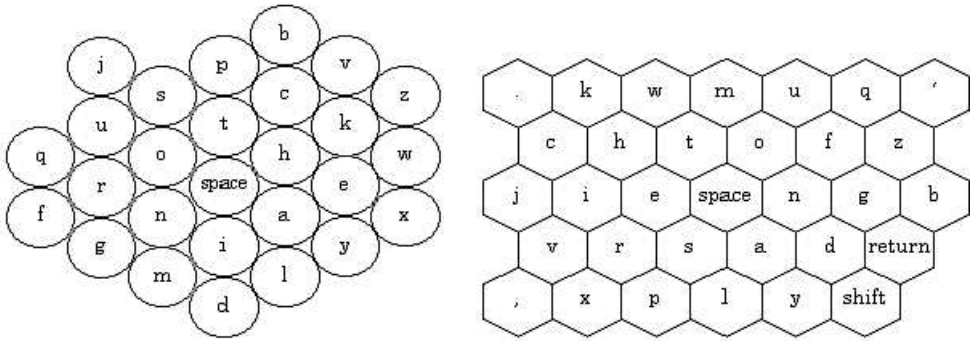


FIG. 17 – Claviers réalisés par techniques inspirées de la physique : A) Clavier Hook ; B) Clavier Metropolis.

Devant l'efficacité de leur méthode Métropolis, Zhai et ses collègues [Smith 01, Zhai 01] ont proposé des modifications dans la manière de calculer l'énergie. Ils ont ainsi voulu prendre en compte deux paramètres supplémentaires ignorés par la loi de Fitts : d'une part, ils ont souhaité faire ressortir de leur clavier l'*alphabetical tuning* qui consiste à essayer de regrouper par zone des caractères proches dans l'alphabet. L'hypothèse sous-jacente était d'aider les utilisateurs novices à se familiariser avec leur nouvelle disposition de touches.

Cette hypothèse a été vérifiée par une expérimentation qui montre une augmentation de 9% des performances de l'utilisateur lorsque celui-ci utilise un clavier généré par Metropolis et qui regroupe par zone les caractères proches alphabétiquement [Smith 01]. D'autre part, ils ont essayé de rapprocher les caractères qui apparaissent dans les mots les plus fréquemment utilisé en anglais. Ces deux apports dans leur fonction d'énergie ont donné naissance au clavier ATOMIK (pour *Alphabetically Tuned and Optimized Mobile Interface Keyboard*). Les performances *a priori* du clavier ATOMIK d'après la formule 9 sont un peu moins bonnes que ceux de Metropolis. Cependant ce clavier est apparemment plus apprécié par les utilisateurs.

L'application de l'algorithme Metropolis ayant apporté de bons résultats de performance de saisie, le clavier ATOMIK a été complété de manière à obtenir un clavier utilisable pour tout type de tâche [Zhai 02a]. L'ajout de touches supplémentaires pour ces nouvelles fonctions a diminué un peu les performances *a priori* que l'on peut avoir avec ce clavier, néanmoins ses performances de saisie (39,9 wpm) restent nettement supérieures à celles d'un clavier QWERTY .

4.4 Bilan

Nous avons présenté dans ce chapitre deux axes de recherche sur la minimisation des mouvements lors d'une tâche de saisie à un pointeur :

- D'une part, la possibilité de regrouper sur une même touche plusieurs caractères. Cette solution au départ envisagée a montré ses limites pour une utilisation par une personne ayant une faible motricité des membres supérieurs.
- D'autre part, seulement améliorer la disposition des touches et l'agencements des caractères sur celles-ci. Plusieurs techniques ont été proposées : de la méthode intuitive et manuelle à l'application de métaheuristiques utilisées pour l'optimisation de problèmes complexes. Ces différentes méthodes ont montré leur efficacité par rapport aux claviers plus classiques tels qu'AZERTY ou QWERTY , à la fois par évaluation *a priori* des performances mais aussi par des expérimentations avec des sujets.

Il reste cependant une famille de métaheuristiques non étudiées pour ce problème. En effet, les métaheuristiques se décomposent en deux grandes familles : les méthodes par voisinage (dont sont issues des méthodes comme le recuit simulé, la recherche tabou) et les méthodes par population dont les méthodes les plus connues sont les algorithmes évolutionnaires et les colonies de fourmis. Selon le problème à résoudre, l'une ou l'autre des

deux familles peut s'avérer plus efficace pour résoudre le problème ou du moins approcher la meilleure solution.

Les méthodes par population ont montré leur efficacité à proposer de bonnes solutions dans des domaines très proches de la saisie à un pointeur. On peut notamment citer les travaux de Levine et Goodenough-Trepagnier sur les algorithmes génétiques pour générer des claviers et claviers ambigus optimisés pour la saisie de textes par des personnes handicapées [Levine 90] ou les travaux plus récents de Eggers et ses collègues sur l'optimisation des caractères d'un clavier physique à l'aide de l'algorithme des colonies de fourmis [Eggers 03].

C'est pourquoi nous proposons maintenant d'appliquer au problème de l'optimisation de la saisie à un pointeur les algorithmes génétiques.

5

Méthode d'optimisation de la disposition des touches et caractères

L'analyse des travaux présentés au chapitre précédent sur la minimisation des mouvements par réorganisation des touches et des caractères (sans revenir sur des claviers ambigus) fait ressortir deux manières d'aborder le problème.

La première solution consiste à associer une touche à chaque caractère. Ainsi lorsqu'on déplace un caractère pour minimiser les distances qui le séparent des autres, on déplace la touche en même temps que le caractère. Cette méthode a été utilisée dans les méthodes inspirées des modèles physiques [Hunter 00, Zhai 00]. Ce procédé a l'inconvénient d'avoir un nombre de solutions infinies. L'espace des solutions ne peut pas être couvert.

Nous avons adopté la seconde approche qui consiste à dissocier le caractère de la touche. Le problème à résoudre est divisé en deux sous problèmes : d'une part, proposer une disposition des touches qui soit la plus compacte possible afin de minimiser les déplacements à effectuer et d'autre part, positionner les caractères sur cette disposition de manière à réduire au maximum les distances entre les caractères les plus fréquemment utilisés.

Nous commencerons ce chapitre par une étude empirique nous permettant d'obtenir une disposition de touches plus compacte. Puis, nous présenterons la manière avec laquelle nous avons appliqué les algorithmes génétiques aux problèmes de l'agencement des caractères.

5.1 Optimisation de la disposition des touches

Afin de minimiser la distance à parcourir avec le pointeur, la première optimisation consiste à organiser les touches de manière à ce qu'elles constituent une disposition la

plus compacte possible. Même si certaines co-occurrences de caractères n'ont lieu que très rarement, il ne faut pas que la distance à parcourir entre les deux caractères soit trop grande. La disposition des touches sur un clavier AZERTY est peu optimale : les touches sont réparties sous la forme d'un rectangle allongé, ce qui donne des caractères assez éloignés comme par exemple 'a' et 'm' ou 'p' et 'w'.

Notre étude vise à trouver une disposition des touches qui soit la plus compacte possible pour diminuer l'amplitude des mouvements sur l'ensemble du clavier. Nous nous sommes focalisés sur l'optimisation pour un clavier de 27 touches rectangulaires et hexagonales (cf Figure 18).

5.1.1 Evaluation d'une disposition de touches

Pour déterminer si une disposition de touches est meilleure ou non qu'une autre, nous avons considéré comme mesure, la distance moyenne qui sépare deux touches quelconques de notre clavier. Cette distance moyenne est définie par :

$$D_{moy.} = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} D_{ij}}{\sum_{i=1}^{N-1} i} \tag{20}$$

où N est le nombre de touches présentes sur le clavier ; D_{ij} représente la distance entre les touches i et j [31] ; $\sum_{i=1}^{N-1} i$ représente le nombre de connexions différentes entre caractères qui existent sur un clavier.

5.1.2 Les dispositions de touches testées

Intuitivement, on peut imaginer qu'une grille compacte - où les touches seraient les moins éloignées en moyenne - aurait une forme de cercle. Notre méthode consiste à tester l'ensemble des possibilités de N touches parmi les M touches d'une grille plus grande pour obtenir le sous-ensemble de N touches le plus compact possible. Nous cherchons un ensemble de 27 touches. Nous avons estimé, vu le peu de touches à chercher, que notre clavier aurait une longueur et/ou une hauteur d'environ six touches. Nous avons considéré une grille d'environ 7×7 touches comme grille de départ. Tester l'ensemble des possibilités est une combinaison C_N^M, par conséquent tester l'ensemble des solutions peut vite devenir insurmontable si N et M sont trop éloignés et M trop grand. Afin de limiter l'ensemble

[31] La distance entre deux touches a été calculée comme correspondante à la longueur du segment ayant comme extrémités les centres des touches.

des solutions, nous avons fixé un noyau de touches au centre de la grille (partie grisée sur la figure 18). Ce noyau étant fixe, cela réduit le nombre de touches à chercher et le nombre de touches à tester, et par conséquent le nombre de solutions possibles.

Nous avons envisagé différents types de grilles (cf. Figure 18) : une où les touches sont organisées en ligne et colonnes (Figure 18 A.) ; une seconde où les lignes sont décalées entre elles d'une demi touche (Figure 18 B.) ; une dernière où les touches sont hexagonales (Figure 18 C.). Les deux derniers types ont l'avantage de proposer plus de voisins directs à une même touche (6 contre 4 sur la grille A.) en raison de leur structure topologique.

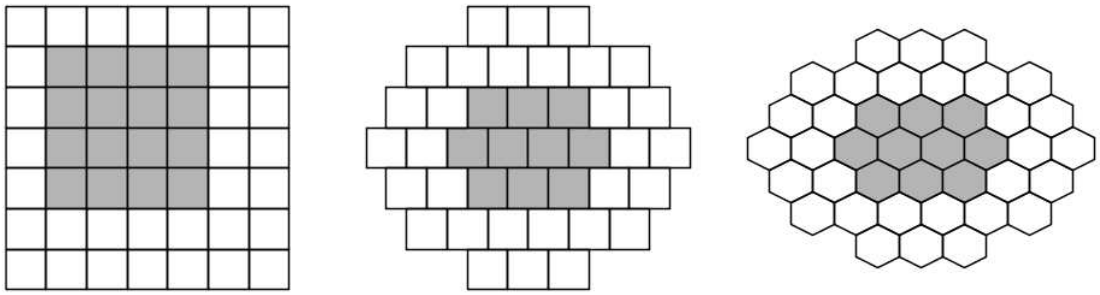


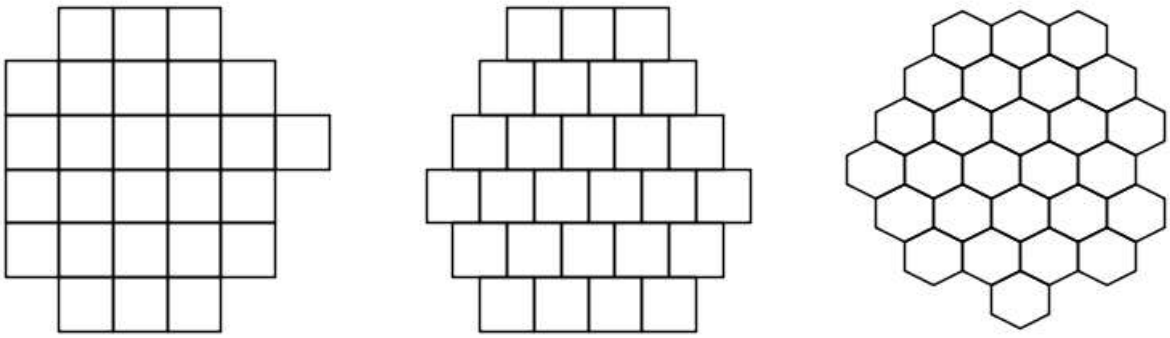
FIG. 18 – Grilles testées pour l'optimisation

5.1.3 Résultats obtenus

La figure 19 illustre respectivement les meilleures dispositions de touches trouvées sur les trois grilles testées. Le tableau 6 donne les distances moyennes pour ces trois dispositions ainsi que celles de claviers logiciels connus de la littérature (cf Chapitre 4) pour être optimisés. Les calculs ont été réalisés en prenant des touches carrées de 34 pixels[32] de côté, et des touches hexagonales inscrites dans un carré de 36 pixels de côté. Nous avons choisi ces valeurs de manière à ce que toutes les touches aient la même surface et qu'ainsi ce facteur influe peu sur le calcul de la distance. Les distances moyennes sont exprimées en pixels. On peut constater qu'il n'existe pas de différences significatives entre les deux propositions de disposition de touches rectangulaires. Par contre, il apparaît que la disposition avec des touches hexagonales diminue de manière assez significative cette distance moyenne. Les résultats de cette dernière sont nettement meilleurs que ceux des

[32]Les chiffres de cette section étant exprimés en pixels, nous les avons arrondis à l'unité la plus proche.

claviers OPTI[33] ou Metropolis par exemple (5,15% de distance en plus à parcourir en moyenne sur la disposition de touches du clavier Metroplis).



A) B) C)

FIG. 19 – Grilles obtenues après optimisation

Nous utiliserons donc comme disposition de touche pour notre clavier final la grille C, puisque c'est elle qui présente la plus faible distance moyenne séparant chaque touche.

Disposition de touche	**Distance moyenne** (en pixels)
AZERTY	116
OPTI I / II	92 / 96
Grille 1	92
Grille 2	92
Metropolis	90
Grille 3	85

TAB. 6 – Distances moyennes des différentes dispositions de touches

Une fois la disposition des touches fixée, l'objectif est de positionner les caractères de manière à ce qu'ils minimisent les mouvements à effectuer lors de la saisie. Cependant pour

[33]OPTI a plus de 27 touches pour représenter plusieurs fois le caractère espace. Pour garder une distance moyenne sur 27 touches, nous avons pris en compte pour chaque touche associée à un caractère que la plus petite des distance la séparant d'une touche espace.

trouver l'agencement de caractères qui minimiserait les mouvements, il serait nécessaire de tester l'ensemble des solutions. En effet, il est impossible de trouver la meilleure solution à ce problème sans les avoir toutes testées : ceci est dû au fait que notre problème est dit "NP-Complet".

Tester l'ensemble des solutions est irréalisable, même avec la puissance de calcul des machines actuelles. Par contre, le fait que notre problème soit NP-Complet nous permet de lui appliquer des méta heuristiques qui trouvent des solutions avoisinant la meilleure solution.

Nous allons dans un premier temps démontrer que ce problème est bien NP-Complet, puis une fois cette hypothèse vérifiée, nous montrerons comment lui appliquer une méta heuristique pour trouver une solution optimale.

5.2 L'agencement des caractères : un problème NP-complet

Pour démontrer qu'un problème est NP-Complet, il est nécessaire de montrer deux propriétés du problème[34] :

- il appartient à la classe des problèmes NP ;
- il s'agit d'un problème NP-Difficile.

5.2.1 Appartenance à la classe NP

Un problème de la classe NP est un problème polynomial non déterministe. Ceci signifie que pour appartenir à cette classe, l'ensemble des solutions du problème doit pouvoir être représenté sous la forme d'un arbre. Chaque chemin de l'arbre représente alors une solution possible du problème. La recherche de la meilleure solution de l'arbre consiste à parcourir l'ensemble de l'arbre.

L'ensemble des solutions de notre problème peut être ramené à un arbre de longueur N et de profondeur M, avec N le nombre de caractères à positionner et M le nombre de touches sur lesquelles on peut positionner un caractère. La figure 20 présente la manière de représenter l'arbre des solutions de notre problème. Chaque noeud au premier niveau de profondeur de notre arbre représente un caractère que l'on peut placer sur la première touche (T_{11}) de notre clavier. Au second niveau de profondeur on retrouve les caractères qui peuvent être positionnés sur la seconde touche (T_{12}) de notre clavier et qui peuvent succéder à leur père : ces caractère doivent être différent de ceux positionnés sur l'ensemble

[34] Définition reprise de [Cormen 94]

des noeuds constituant ce chemin jusqu'à la racine. Ce procédé se reproduit jusqu'à avoir affecté un caractère par touche T_{ij} de notre clavier.

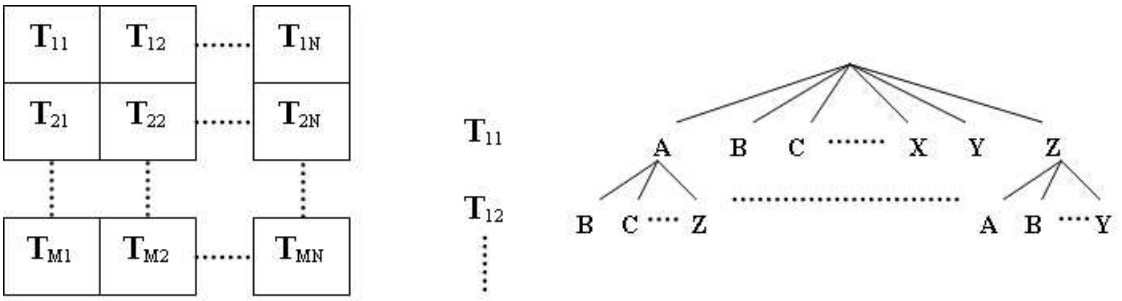


FIG. 20 – Représentation du problème sous forme d'arbre de solutions

Une solution doit pouvoir être calculée avec une complexité polynomiale. Ceci est le cas de notre problème car une solution est calculée grâce à la formule 9. Cette formule est de complexité $O(N^2)$ et donc polynomiale. Notre problème fait donc partie des problèmes de la classe NP.

5.2.2 Problème NP-difficile

Montrer qu'un problème est NP-difficile revient à montrer que :

1. Un problème NP-Complet connu peut être ramené à notre problème ;
2. La transformation de ce problème connu au notre est polynomiale ;
3. La fonction de coût de notre problème est au moins aussi complexe.

On peut ramener le problème du voyageur de commerce à notre problème d'optimisation des caractères. Le problème du voyageur de commerce consiste à rallier un ensemble de villes d'une carte en ne passant qu'une et une seule fois par chacune d'elles et à revenir au point de départ. Connaissant la distance séparant chaque ville, le but est de trouver le chemin qui minimise la distance totale à parcourir.

Dans notre cas, chaque ville correspond à un caractère. L'ordre de passage dans les villes correspond à l'ordre dans lequel on positionne les caractères sur les touches. Par exemple, le passage à la ville V_i en j^e position se transforme par le positionnement du caractère C_i sur la touche T_j. Dans le cas du voyageur de commerce, la fonction à minimiser ne prend en compte que les transitions entre les villes qui se succèdent. Dans notre cas, selon le chemin pris, la valeur de chaque transition change et notre fonction d'évaluation prend en compte l'ensemble des transitions entre deux caractères et pas simplement celles empruntées par le chemin choisi. Notre problème est donc NP-Complet.

Il est par conséquent impossible de faire une recherche exhaustive de la meilleure solution dans l'arbre des solutions. Nous allons voir comment résoudre notre problème par l'utilisation d'algorithmes génétiques qui sont une des meilleures méthodes de résolution de problème NP-Complet.

5.3 Méthode d'optimisation par algorithme génétique

5.3.1 Le principe de base

Les algorithmes génétiques font partie de la classe des algorithmes évolutionnistes. Ils ont été étudiés par Holland [Holland 75] et De Jong [Jong 75] au milieu des années 70 à l'Université du Michigan. Cette technique d'optimisation, jusque-là très théorique, a ensuite été reprise et vulgarisée par Goldberg en 1989 pour la résolution de problèmes complexes [Goldberg 89].

Le but de ces algorithmes est de simuler le processus de sélection naturelle[35] dans un environnement hostile où les individus se reproduisent de manière à pouvoir y survivre. L'environnement hostile est représenté par le problème à résoudre où chaque individu de cet environnement est une solution potentielle du problème. La reproduction des individus est réalisée de façon à se rapprocher de la solution du problème.

A la différence des autres algorithmes évolutionnistes[36] [Bäck 93], les algorithmes génétiques sont considérés comme une méthode de résolution de problème "ascendante" : c'est-à-dire que la solution optimale est construite à partir de parties des meilleurs individus de la population précédente. Pour converger plus rapidement vers un individu optimal, les meilleurs individus auront plus de chance de se reproduire pour former une nouvelle population d'individus. A l'inverse, les moins bons ne seront que très peu utilisés lors de la reproduction entre individus, ainsi leurs moins bonnes parties tendent à disparaître avec l'évolution de la population.

Les opérations successives utilisées dans les algorithmes génétiques sont illustrées sur la figure 21. La première population G_0 est créée aléatoirement. Celle-ci contient N individus $I_{n\in[1\cdots N]}$. Parmi eux, seuls les $N/2$ meilleurs sont gardés après la phase de sélection. Ces derniers se reproduisent entre eux avec une plus forte probabilité de reproduction pour les meilleurs d'entre eux. De cette phase de reproduction (c'est-à-dire croisement et mutation), une nouvelle population est obtenue. On réitère alors les mêmes étapes de

[35] Tout cela s'inspire des théories de l'évolution proposées par Darwin [Darwin 59]

[36] Il existe aussi les stratégies évolutionnistes [Rechenberg 73, Schwefel 77] et la programmation évolutionniste [Fogel 66].

sélection et de reproduction sur celle-ci. L'algorithme s'arrête lorsqu'on considère avoir convergé vers une solution suffisamment optimale.

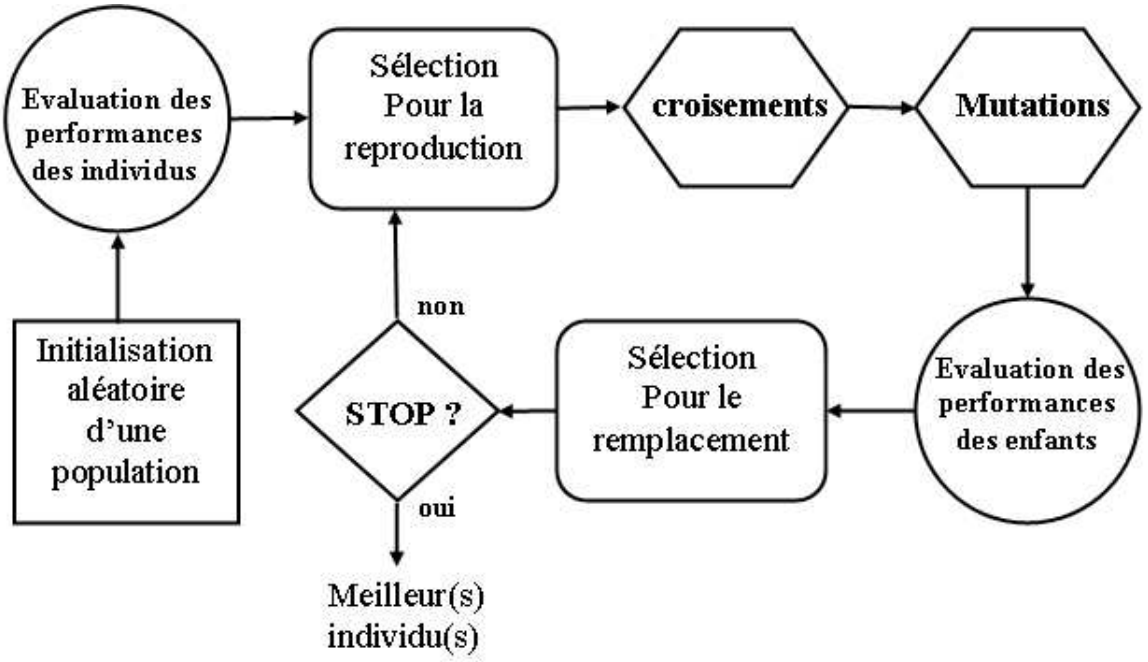


FIG. 21 – Principe d'un algorithme génétique (repris de [Dréo 03])

5.3.2 La reproduction

La reproduction se produit à partir de deux opérations successives :

- **Le croisement** qui s'effectue entre deux individus sélectionnés. Ces derniers créent un nouvel individu à partir des meilleures parties qui les constituent ;
- **La mutation** qui a lieu sur les individus qui sont créés lors de la phase de croisement. Elle consiste à modifier au hasard certaines parties de l'individu.

5.3.2.1 Le croisement entre deux individus

On trouve principalement deux types de croisement dans la littérature :

- **Le croisement simple** qui consiste à découper en deux morceaux et au même niveau les deux individus parents, puis à constituer deux nouveaux individus contenant chacun un morceau de chaque parent ;
- **Le croisement multiple** qui fonctionne de la même manière que le croisement simple, sauf que les individus parents sont cette fois-ci coupés en plusieurs morceaux (au

même endroit sur chaque parent), et les deux nouveaux individus créés sont constitués à partir de morceaux des deux individus parents.

5.3.2.2 La mutation

La mutation est réalisée sur le nouvel individu issu du croisement. Celle-ci peut intervenir sur n'importe quelle partie de l'individu. Elle a pour but de modifier légèrement l'individu. Ce changement n'aurait peut-être jamais eu lieu avec la reproduction. Ainsi cela peut permettre d'ouvrir un nouvel espace de recherche, et par conséquent, proposer par la suite une meilleure solution.

5.4 Architecture utilisée pour l'optimisation de l'agencement des caractères

La figure 22 présente l'architecture que nous avons utilisée pour appliquer les algorithmes génétiques à notre problème. Nous l'avons conçue de manière à ce que notre approche soit la plus générique possible pour pouvoir :

- tester, par la suite, d'autres méta heuristiques pour l'optimisation,
- mais aussi modifier facilement certaines variables nécessaires à la génération du clavier.

Ainsi, il est possible dans notre système de changer la disposition des touches, l'alphabet des caractères à associer à celles-ci. Ces deux caractéristiques sont données à l'algorithme génétique à l'initialisation de la première population.

Comme nous l'avons vu sur la Figure 21, deux phases de sélection ont lieu lors des différentes itérations de l'algorithme génétique. Pour l'évaluation de nos dispositions de caractères, nous avons utilisé comme fonction de coût la formule 9 de Soukoreff et MacKenzie qui estime *a priori* les performances d'un utilisateur avec le clavier généré. L'objectif est ici de maximiser cette fonction. Les connaissances linguistiques sont représentées par une table de bi grammes contenant l'ensemble des fréquences d'apparition des différentes co-occurrences de caractères. Nous avons choisi de séparer cette table de bi grammes de la fonction d'évaluation afin de pouvoir changer facilement cette table. Le changement de la table de bi grammes permet ainsi de proposer des dispositions de caractères optimisées pour différentes langues ou différents contextes.

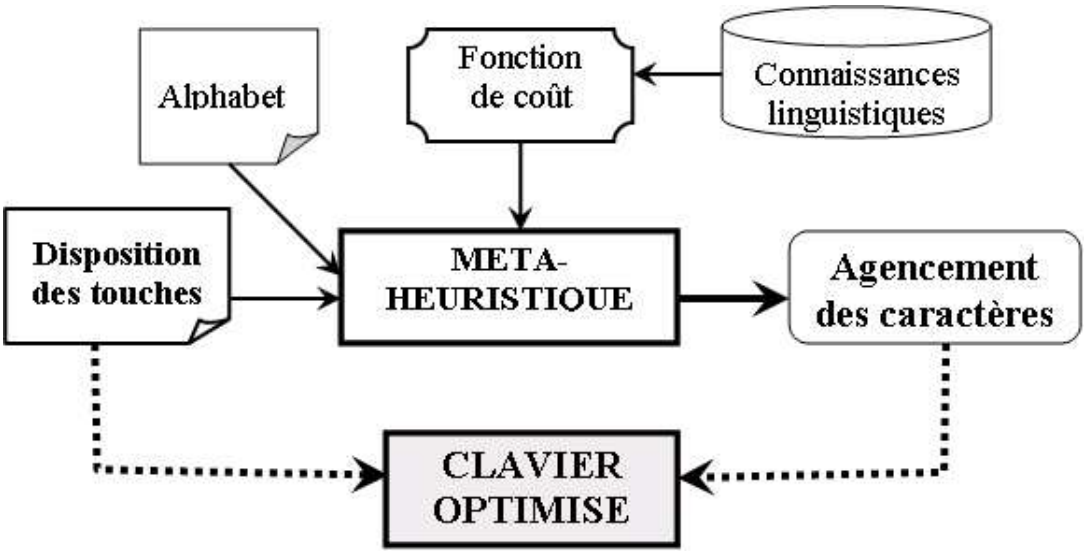


FIG. 22 – Architecture utilisée pour l'optimisation de l'agencement des caractères

5.4.1 Tranposition des algorithmes génétiques à l'agencement des caractères

A la base des algorithmes génétiques, les individus étaient codés sous forme de chaînes de bits. Les croisements et les mutations avaient lieu sur une partie de cette chaîne. Depuis quelques années, les domaines d'applications pour les algorithmes génétiques se sont étendus et les techniques de codage ont évolué. L'individu peut maintenant être représenté sous forme d'objet [Khuri 95, Reeves 95], de matrice [Caux 95] ou de liste ; par exemple une liste de villes à parcourir pour le problème du voyageur de commerce [Grefenstette 87, Lin 93].

Dans notre cas, chaque individu est représenté par une liste ordonnée de caractères où chaque position dans cette liste représente une touche particulière sur le clavier (cf Figure 23). La phase d'initialisation de la première population consiste pour chaque individu à placer aléatoirement l'ensemble des caractères de l'alphabet donné à une position dans cette liste ordonnée. La sélection des meilleurs individus se fait grâce à la fonction de coût décrite à la section 5.4. Les itérations sont arrêtées lorsque l'individu présentant les meilleures performances de la population est identique pendant un certain nombre d'itérations consécutives.

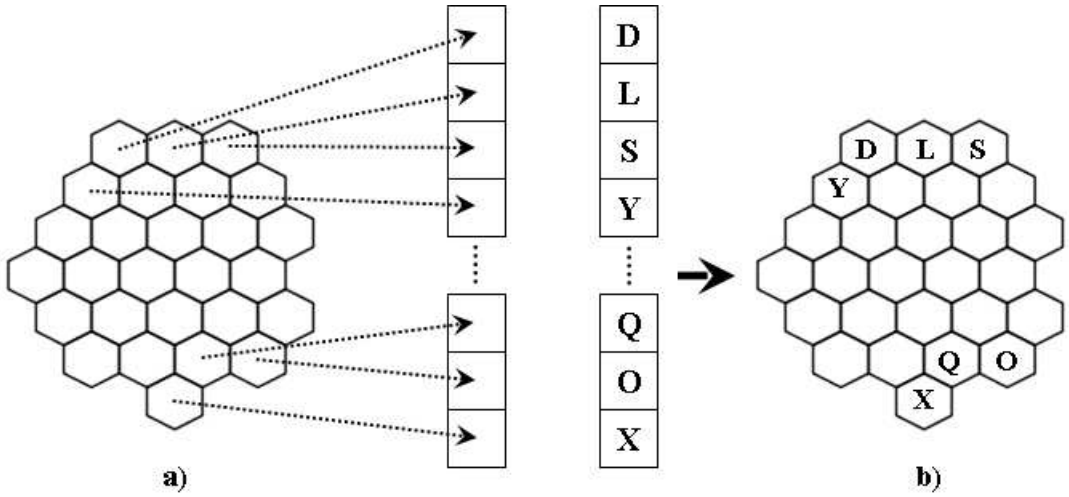


FIG. 23 – Représentation de notre problème pour l'application des algorithmes génétiques : a) Chaque touche (définie par sa forme, sa taille et sa position sur le clavier) est affectée à une position dans la liste ; b) Exemple d'individu et de sa transposition sous forme de clavier.

5.4.2 Potentiel d'amélioration

Afin de converger plus rapidement vers un individu ayant des performances optimales, il est possible de sélectionner les parties que l'on veut garder pour la reproduction. Pour pouvoir sélectionner les "meilleurs" caractères d'un clavier, nous avons défini le potentiel d'amélioration.

Le potentiel d'amélioration d'un caractère consiste à calculer la différence qui existe entre la "performance" qu'il possède sur le clavier donné et celle qu'il pourrait avoir au maximum.

5.4.2.1 La performance d'un caractère

La performance d'un caractère i consiste à calculer le nombre de secondes qui seraient nécessaires pour pouvoir saisir tous les caractères à partir du caractère i. Ce calcul est basé sur la loi de Fitts, et fortement inspiré de la formule 9 :

$$t^i = \sum_{\substack{j=1 \\ i \neq j}}^{27} \frac{P_{ij}}{IP} \left[log_2 \left(\frac{D_{ij}}{W_j} + 1 \right) \right] \tag{21}$$

avec :

- t^i : la performance du caractère i ;
- P_{ij} : la fréquence d'apparition du caractère j après le caractère i ;
- IP : l'indice de performance. Nous avons choisi de prendre la même valeur de IP que celle utilisée dans la fonction d'évaluation ;
- D_{ij} : la distance séparant les touches associées aux caractères i et j ;
- W_j : la taille de la touche associée au caractère j.

5.4.2.2 La performance maximale d'un caractère

La première étape consiste à calculer la performance maximale de chaque caractère pour la disposition de touches donnée. Nous calculons cette performance maximale en plaçant le caractère i sur la touche la plus centrale du clavier : c'est-à-dire celle qui est en moyenne à la plus petite distance de chacune des autres touches. Les autres caractères sont positionnés sur les autres touches en fonction de leur fréquence d'apparition après le caractère i : le caractère le plus fréquemment utilisé après le caractère i sera positionné sur la touche la plus proche, et ainsi de suite jusqu'à ce que le caractère le moins utilisé après le caractère i soit positionné sur la touche la plus éloignée. Une fois les caractères positionnés sur le clavier, nous calculons alors la performance du caractère en question. La valeur obtenue sera considérée comme la performance maximale que peut avoir ce caractère. Cette première étape est réalisée à l'initialisation de l'algorithme. La performance maximale de chaque caractère est valable pendant tout l'algorithme.

Ensuite, lors de chaque itération, pour chaque individu I_n, nous calculons la performance de chacun des caractères tels qu'ils sont placés sur l'individu. Pour uniformiser les valeurs, nous calculons le potentiel d'amélioration de chaque caractère : c'est-à-dire la différence de performances qui existe entre sa performance sur l'individu I_n et sa performance maximale :

$$p^i_{I_n} = t^i_{max} - t^i_{I_n} \tag{22}$$

Le classement des lettres se fait à partir des valeurs de $p^i_{I_n}$. Ainsi, plus une valeur est petite, plus sa performance sur l'individu est proche de sa performance maximale, et moins le caractère a besoin d'être déplacé.

5.4.3 Application de la reproduction

5.4.3.1 Le croisement entre individus

De la façon dont nous avons codé les individus pour notre problème, les croisements tels que présentés à la section 5.3.2, ne sont pas applicables car ils risqueraient de produire des individus où tous les caractères ne seraient pas présents et inversement, certains caractères seraient redondants. C'est pourquoi nous appliquons la reproduction de la manière suivante :

1. Pour chaque individu parent, un classement des caractères est effectué en fonction de la pertinence de leur position sur le clavier ;
2. A partir des deux individus parents, un nouvel individu est créé. Le but est de replacer aux mêmes positions sur le nouvel individu les caractères (des deux individus parents) ayant de faibles potentiels d'amélioration. Pour cela, ils placent tour à tour un caractère sur le nouvel individu, en prenant les caractères dans l'ordre décroissant du classement et en vérifiant que le caractère à positionner n'a pas déjà été positionné par l'autre individu et si la position où ils veulent le mettre n'a pas déjà été prise avant. Dans le cas contraire, ils prennent le caractère suivant dans le classement ;
3. Une fois que les deux individus ont positionné les caractères tour à tour, il est possible qu'il reste des caractères qui n'aient pas été placés, et qu'il reste des positions libres sur le nouvel individu. Dans ce cas-là, les caractères restant à positionner sont placés au hasard sur les cases restantes.

Afin d'illustrer ces différentes étapes, prenons l'exemple concret présenté à la figure 24. Les individus parents sont les deux listes de caractères grisées. Nous appellerons I_1 le parent de gauche et I_2 le parent de droite.

Les deux individus commencent par faire un classement des lettres. Imaginons que le classement de I_1 soit {G, D, C, A, F, B, E} et le classement de I_2 soit {A, F, C, B, E, G, D}. Les deux individus parents positionnent ensuite leurs meilleurs caractères sur le nouvel individu (cf Figure 24 a.) : I_1 place le 'G' en 3^e position sur le nouvel individu ; à son tour I_2 positionne 'A' ; I_1 essaye de positionner 'D' en 4^e position mais I_2 ayant déjà positionné le 'A' à cette position, I_1 ne peut placer le 'D' et passe directement au positionnement de 'C' ; I_2 positionne lui le 'B' (il ne peut positionner le F car la place est déjà prise et le 'C' a déjà été positionné par I_1). Les opérations se succèdent ainsi jusqu'à ce que chaque individu ait essayé de positionner tous ces caractères.

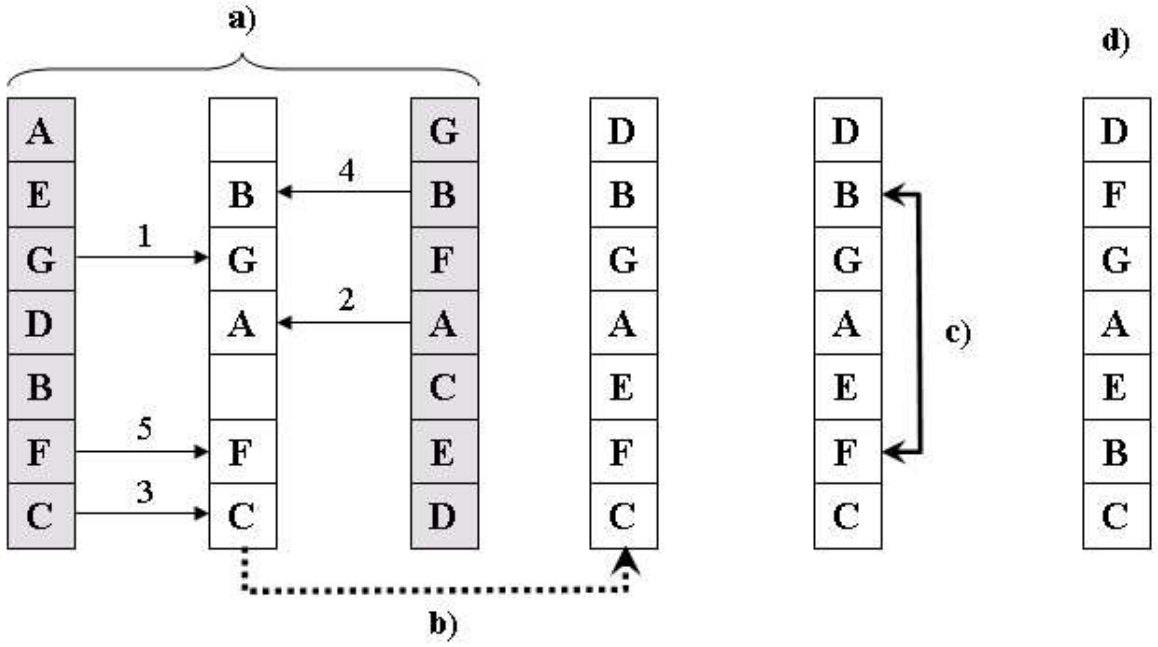


FIG. 24 – Les différentes étapes de la reproduction : a et b) Reproduction ; c) Mutation ; d) Individu obtenu à la fin de la reproduction.

A la fin de cette étape, on peut voir que le nouvel individu n'est pas complet. Les caractères manquants (sur l'exemple 'D' et 'E') sont alors positionnés de manière aléatoire sur les positions restantes (étape b. sur la Figure 24).

5.4.3.2 La mutation

Dans notre cas, la mutation consiste à interchanger les positions entre 2 caractères (cf. Figure 24 c.). Comme nous l'avons dit, la mutation peut intervenir partout. Nous ne classons plus ici les lettres en fonction de leur performance. Ainsi une lettre avec un très bon potentiel peut être déplacée alors qu'elle ne l'aurait jamais été juste par croisement.

A la fin de la mutation, nous obtenons donc un individu prêt à être intégré à la nouvelle population d'individus (cf. Figure 24 d.).

La reproduction, telle que nous l'avons conçue, implique de pouvoir classer les lettres. Pour cela, nous avons défini le calcul d'une valeur représentant le potentiel d'amélioration pour chaque caractère.

L'application des algorithmes génétiques nous permet de générer des agencements de caractères pour une disposition de touches donnée. Chaque génération d'agencement

de caractères associé à une disposition de touche est appelée clavier GAG (Généré par Algorithmes Génétiques).

6

Génération de claviers GAG : résultats et comparaisons

Pour prouver la validité de notre méthode, nous proposons de générer des claviers optimisés pour la langue anglaise. Ceci me permettra de comparer les résultats obtenus avec notre méthode à ceux des claviers reconnus pour avoir été optimisés (FITALY, OPTI et Metropolis) ainsi qu'aux résultats du clavier de référence (QWERTY).

Nous l'appliquerons ensuite pour le français en utilisant les mêmes conditions. Seule la table de bigrammes représentant les caractéristiques de la langue change.

6.1 Comparaison des performances avec les claviers de la langue anglaise

Cette comparaison avec les claviers anglais s'est déroulée en deux phases :

- dans un premier temps, nous avons souhaité montrer l'efficacité des algorithmes génétiques pour générer une disposition de caractères sans chercher à optimiser la disposition des touches ;
- puis, nous avons cherché à vérifier si le fait d'optimiser la disposition des touches avait une importance sur la génération du clavier.

6.1.1 Evaluation des algorithmes génétiques

Pour évaluer seulement la production des algorithmes génétiques, c'est-à-dire l'agencement des caractères, nous avons pris comme point d'entrée de nos algorithmes génétiques

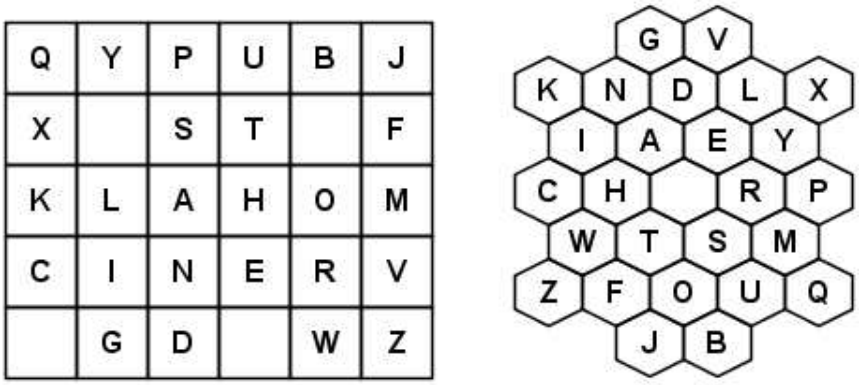
les dispositions de touches utilisées pour les claviers connus. Nous avons ainsi testé sur deux dispositions de touches :

- La première qui reprend les caractéristiques du clavier OPTI II, c'est-à-dire 30 touches rectangulaires réparties en cinq lignes et six colonnes. En plus de la disposition des touches, nous avons aussi repris le même alphabet, à savoir les 26 lettres de l'alphabet et surtout quatre caractères espace à répartir sur le clavier ;
- La seconde sur le principe du clavier Metropolis avec des touches hexagonales et un alphabet de 27 caractères.

Ces deux premiers claviers (nommés respectivement GAG I et GAG II) ont été générés en utilisant 20 000 individus par population. L'algorithme était arrêté après 500 itérations successives où l'individu présentant les meilleures performances n'évoluait pas. Nous avons utilisé les formules 9 et 14 de Soukoreff pour calculer la limite supérieure des performances théoriques que pourrait avoir un utilisateur avec un clavier. Nous avons utilisé comme connaissance linguistique la table de bigrammes de Maizner et Tresselt [Mayzner 65].

Nous avons calculé ensuite les distances entre les touches en prenant la longueur de la droite ayant comme extrémités le centre des touches concernées. Nous avons choisi de prendre la largeur des touches pour W. Enfin nous avons pris comme valeurs pour les variables de la formule 9 : $a = 0$, $b = 4,9$ et $MT_{repeat} = 0,127$. Nous avons repris les valeurs suggérées par [MacKenzie 99a, Zhai 00].

La figure 25 présente les claviers obtenus à partir des caractéristiques données, et après application des algorithmes génétiques.



A) GAG I B) GAG II

FIG. 25 – Claviers générés par algorithmes génétiques pour l'anglais

Le tableau 7 résume les résultats obtenus lors de notre évaluation. Nous avons réévalué les claviers QWERTY, FITALY, OPTI et Metropolis avec les mêmes paramètres de manière

à avoir une évaluation comparative de tous les claviers. Les performances sont données en mots par minute et l'apport donne l'augmentation en pourcentage des performances des claviers optimisés par rapport au clavier QWERTY .

Clavier	Performances *(wpm)*	Apport
QWERTY	29.91	-
FITALY	40.96	36.94 %
OPTI II	42.82	43.16 %
GAG I	45.57	52.36 %
Metropolis	45.11	50.82 %
GAG II	46.43	55.23 %

TAB. 7 – Performances des différents claviers optimisés pour l'anglais.

Ainsi, nous pouvons constater que les algorithmes génétiques apportent de meilleurs résultats sur disposition équivalente de touches que les techniques qui avaient été testées jusqu'à présent. Le gain est plus important pour GAG I par rapport au clavier OPTI II qui a les mêmes caractéristiques de touches et d'alphabet (6,42% de gain pour GAG I par rapport à OPTI II). Ce dernier avait été réalisé intuitivement.

Il est intéressant de souligner que le clavier GAG I a de meilleures performances que le clavier Metropolis dont les caractéristiques des touches devraient lui être plus favorables. Ceci m'amène à formuler l'hypothèse que l'agencement des caractères a plus d'importance dans le calcul des performances que la disposition des touches. Toutefois, à technique d'optimisation identique, la disposition hexagonale apporte de meilleurs résultats que la disposition rectangulaire : GAG II a quand même de meilleures performances que GAG I.

Les algorithmes génétiques prouvent à travers ces résultats qu'ils s'appliquent avec succès au problème de l'agencement des caractères. Le clavier Metropolis -qui était jusque-là le clavier avec le meilleur agencement de caractères connu- a des performances théoriques de saisie légèrement plus basses que celles des algorithmes génétiques (GAG II obtient des performances théoriques supérieures de 2,92% par rapport à celles de Metropolis).

Avant de proposer un clavier optimisé pour le français, regardons s'il est possible d'améliorer ces performances sur une disposition de touches plus compacte.

6.1.2 Application sur une disposition de touches optimisée

Nous avons appliqué à la disposition de touches, réalisée à la section 5.1.3, les algorithmes génétiques. Pour cette génération, nous nous sommes basés sur une population de 25 000 individus à chaque itération. Les itérations n'ont été arrêtées qu'après avoir eu 1000 cycles avec les mêmes performances pour le meilleur individu. La figure 26 présente le clavier obtenu.

FIG. 26 – GAG III : Clavier généré par algorithme génétique pour l'anglais avec une disposition de touche plus compacte

Par rapport aux claviers générés à la section précédente, nous avons choisi ici de prendre W' (cf. Figure 4) comme taille de la cible à atteindre. Ce choix s'explique par la forme non rectangulaire des touches. Zhai et ses collègues [Zhai 00] pour le même type de touche, ont proposé de prendre le cercle inscrit dans la touche. Cette approximation donne alors à la touche la même taille quelle que soit la direction avec laquelle on arrive au moment du pointage. Selon la direction d'où on arrive, la touche sera plus ou moins grande. Par exemple, sur une touche hexagonale comprise dans un carré de 36 pixels de coté, selon l'angle d'arrivée sur la touche, la taille de celle-ci peut varier de 32 à 40 pixels, soit un écart de 20%. Cet écart peut avoir son importance au moment de positionner les caractères.

Avec ce nouveau principe de calcul de la taille des touches à atteindre, nous avons recalculé les performances des claviers Metropolis et GAG II pour pouvoir les comparer avec celles du nouveau clavier généré (GAG III). Le tableau 8 présente les résultats de ces trois claviers. Les claviers générés par algorithmes génétiques sont meilleurs que les deux proposés par l'algorithme de Metropolis. L'écart n'est cependant pas très significatif, mais nous permet de dire que notre méthode est efficace.

Nous pouvons aussi remarquer que les deux claviers GAG II et GAG III sont très proches en terme de performance. La disposition des touches optimisée n'a que très peu

d'impact par rapport à une disposition plus classique de touche. Il apparaît par conséquent plus important de prendre en compte l'agencement des caractères plutôt que la disposition des touches.

Clavier	Performances *(wpm)*	Apport
Metropolis	43.21	-
Metropolis II	43.68	-
GAG II	44.81	3.7 %
GAG III	44.87	3.84 %

TAB. 8 – Performances des différents claviers optimisés pour l'anglais.

6.2 Proposition d'un clavier optimisé pour le français

La génération de clavier par algorithme génétique ayant montré de bons résultats pour la langue anglaise, nous avons appliqué ce même algorithme pour le français. Nous avons utilisé la même méthode que pour la génération du clavier GAG III : a savoir, nous avons repris la meilleure disposition de touches et appliqué la même fonction d'évaluation en prenant W' comme taille des touches. Pour que ce clavier soit optimisé pour la saisie en français, nous avons utilisé une table de bigrammes générée à partir de la base de données de ressources lexicales du français BDLex [De Calmès 98].

FIG. 27 – Clavier généré par algorithme génétique pour le français

La figure 27 présente le clavier généré pour le français. Ce dernier a des performances théoriques de l'ordre de $45,84$ mots par minute si l'on considère qu'un mot est formé de

cinq caractères comme pour la langue anglaise. A titre de comparaison, le clavier AZERTY a des performances de 30, 76 mots par minute. Notre clavier GAG optimisé pour le français apporterait donc en théorie un gain en vitesse de saisie de texte de 49% par rapport au clavier logiciel AZERTY, et pour des utilisateurs experts.

6.3 Evolution des performances au cours des itérations

La figure 28 représente la courbe d'évolution de la meilleure performance au fur et à mesure des cycles de reproduction du clavier nommé GAG III. Nous pouvons ainsi constater que la courbe est de type logarithmique. La meilleure performance est approchée assez rapidement (environ 200 cycles). Ensuite, le clavier n'est souvent que très légèrement amélioré. Nous constatons qu'au delà de 900 itérations les performances sont quasiment stables.

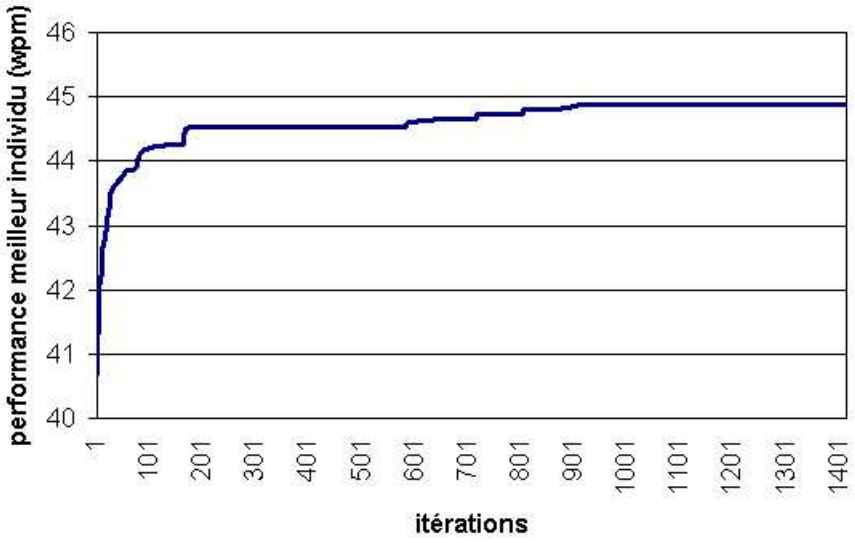


FIG. 28 – Evolution du meilleur individu au cours des itérations

Conclusion

Nous avons présenté dans cette partie une application des algorithmes génétiques au problème de l'agencement des caractères. Celle-ci s'est révélée être aussi efficace dans la recherche d'une solution optimale que les meilleures techniques qui avaient été testées jusque-là.

Les résultats obtenus avec cette méthode sont légèrement meilleurs que ceux présentés jusque-là. Devant la stagnation des performances autour d'un seuil (environ 43 mots par minute) pour les claviers anglais, on peut penser que notre méthode ainsi que Metropolis proposent des claviers proches du clavier optimal. Il serait étonnant de trouver un clavier bien meilleur que ceux là. Cette méthode nous a permis de générer un clavier dont l'agencement des caractères est optimisé pour le français, chose qui n'existait pas jusqu'à présent.

Néanmoins, quelle que soit la manière d'agencer les caractères, certains seront toujours éloignés par rapport à d'autres. Même si les distances sont minimisées en moyenne sur les bigrammes les plus fréquents d'une langue, certains bigrammes moins fréquents nécessiteront de parcourir une distance plus importante pour pouvoir la saisir.

Nous avons démontré que l'optimisation de la disposition des touches est un facteur moins important que l'agencement des caractères. Le gain apporté à disposer les touches de manière à ce qu'elles soient à une distance moyenne faible peut être compensée par un agencement des caractères optimisé.

Perspectives

Dans la loi de Soukoreff, basée sur la loi de Fitts, deux paramètres importants entrent en jeu pour prédire les performances de l'utilisateur avec un clavier donné : la distance à parcourir entre chaque caractère à saisir, mais aussi la taille de la touche sur laquelle se trouve le caractère à saisir. Or, jusqu'à présent, toutes les propositions de maximisation de la vitesse de saisie de textes se sont uniquement focalisées sur la minimisation des distances entre les touches.

Les recherches menées sur la minimisation des distances atteignent semble-t-il leur limite. Je pense qu'il serait intéressant de se pencher sur la question de la taille des touches. Pourquoi un caractère fréquemment utilisé ne serait-il pas associé à une touche plus grande ? On pourrait aussi envisager un clavier avec une disposition de touches où toutes n'auraient pas la même taille (comme par exemple sur la Figure 59). On pourrait alors appliquer les mêmes méthodes que celles que nous venons de voir pour la minimisation des distances.

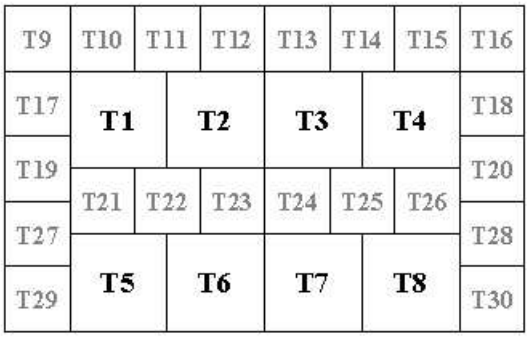


FIG. 29 – Disposition de touches de différentes tailles

Troisième partie

Claviers augmentés

Introduction

La seconde partie de ce travail de thèse a concerné nos travaux relatifs à l'optimisation de la disposition des touches et des caractères. Celle-ci une fois générée est fixe pour la tâche de saisie de textes.

Nous allons maintenant proposer d'autres méthodes d'optimisation des claviers logiciels par l'ajout de comportement dynamique au cours de la saisie de textes. Dans la suite de ces travaux, nous nommerons ces claviers "clavier augmenté". Celui-ci est un clavier qui utilise de manière directe ou indirecte des éléments ou des fonctionnalités supplémentaires qui visent à apporter une aide à la saisie. Cette aide s'appuie souvent sur un système de prédiction. Cette prédiction peut avoir lieu au niveau des caractères ou des mots. Grâce à celle-ci, plusieurs systèmes de saisie de textes augmentés ont été conçus pour profiter pleinement de cette prédiction. Le lecteur trouvera un excellent état des lieux des systèmes de prédiction dans [Boissière 06] et [Schadle 03].

Avant de présenter nos travaux, nous faisons, dans cette introduction, une revue des différents types de recherches qui ont abouti à l'élaboration de claviers logiciels augmentés.

FIG. 30 – KeyStrokes : Exemple de clavier augmenté par une liste de mots les plus probables [Bérard 04a]

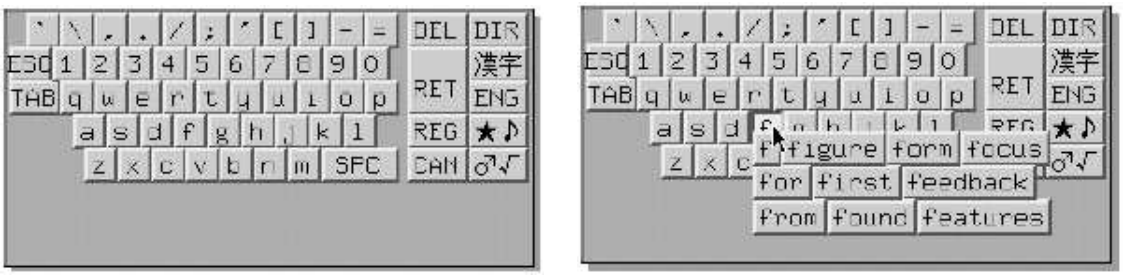
Une des utilisations les plus classiques d'un système de prédiction consiste à proposer un ensemble de mots les plus probables par rapport au début de la saisie de l'utilisateur. Dans le cas où un des mots proposés est celui recherché par l'utilisateur, celui-ci peut alors valider ce mot au lieu de finir de le saisir via les touches du clavier. Ceci a pour but de diminuer le nombre de caractères à saisir. Plusieurs systèmes ont été proposés sur ce principe. On peut notamment citer les systèmes KeyStrokes[37] [Bérard 04a, Niemeijer 05] (cf. Figure 30), HandiAS [Pévédic 97, Maurel 01], Reactive Keyboard [Darragh 90], SybiMot [Schadle 03] ou FASTY [Beck 04].

Bien que ces systèmes diminuent généralement de manière assez significative le nombre de caractères à saisir, la position de la liste proposant les mots n'est pas toujours optimale. Celle-ci est, soit placée sur un des côtés du clavier, comme par exemple pour KeyStrokes, soit elle recouvre une partie du clavier comme proposé par Masui dans son système POBox [Masui 98] (cf. Figure 31). Ces deux méthodes ont leurs avantages et inconvénients :

- Dans le premier cas, le fait que la liste soit à côté permet de ne pas cacher le clavier. L'utilisateur peut alors continuer à saisir sans se servir de la liste, sans qu'il soit gêné par celle-ci. Par contre, s'il veut sélectionner un des mots de la liste, cela implique un mouvement avec une amplitude plus ou moins longue à réaliser pour l'atteindre. En plus de ce mouvement, le fait que la liste soit décalée par rapport au clavier entraîne une attention particulière pour regarder les mots qui composent cette liste. La perte de temps associée à la recherche du mot dans la liste, ainsi que celui de pointage de l'item associé au mot dans la liste, peut faire perdre plus de temps à l'utilisateur que ce qu'il a gagné. Toutefois Bérard et Niemeijer [Bérard 04b] ont montré l'influence de la longueur de la liste de prédiction : avec une liste de 5 mots le rendement est de 46%, avec 10 mots de de 50,8%, et avec 15 mots de 53,1%.
- Dans le second cas, le mouvement et le focus d'attention supplémentaire sont minimisés car la liste se trouve à proximité du dernier caractère saisi. Cependant cette proximité masque une partie du clavier (cf. Figure 31 B)), ce qui peut gêner l'utilisateur dans sa saisie si celui-ci n'est pas intéressé par les mots proposés dans la liste.

Quant au système VITIPI [Boissière 90], celui-ci affiche à l'écran la chaîne de caractères prédits. L'utilisateur a alors le choix de valider cette chaîne de caractères, de choisir parmi une liste ou alors de continuer la saisie pour avoir une autre liste de prédiction. Ce système a l'avantage de pouvoir fonctionner avec tout type de système de saisie de texte : aussi bien un clavier logiciel qu'un clavier physique. L'autre avantage de VITIPI

[37] http ://www.assistiveware.com/keystrokes.php



A) B)

FIG. 31 – Exemple de clavier augmenté par une liste de mots prédits : le clavier POBox [Masui 98]

est qu'il est également capable de s'auto-adaptater en fonction de la saisie de l'utilisateur [Boissière 06].

Ces systèmes basés sur la prédiction de mots ont l'avantage de diminuer le nombre de caractères à saisir. Cependant, mis à part dans des contextes bien précis, ce type de prédiction n'améliore pas la vitesse de saisie. Une autre technique consiste alors à optimiser le clavier dynamiquement tout au long de la saisie en faisant intervenir un système de prédiction de caractères. Nous allons maintenant montrer les différentes utilisations que l'on peut en faire.

L'objectif n'est plus ici de diminuer le nombre de caractères à saisir, mais de faciliter la saisie de textes, d'augmenter la vitesse de saisie ou encore de diminuer le taux d'erreur de saisie.

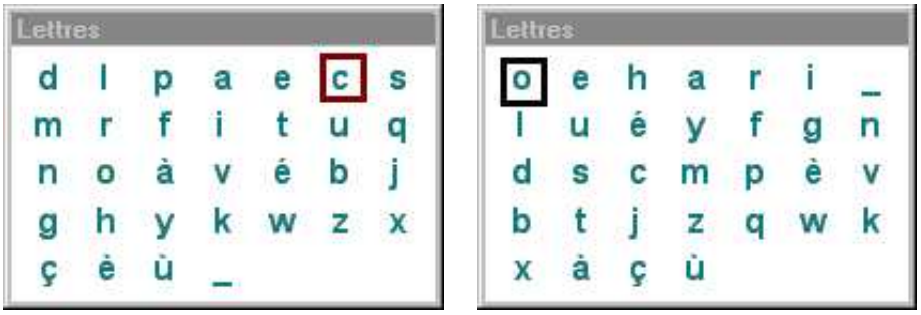
On peut, par exemple, citer les travaux sur les indices visuels [Magnien 04, Gong 05]. Ces travaux ont pour but de faciliter la recherche (sur un clavier logiciel) du caractère à saisir pour un utilisateur novice. Après chaque caractère saisi, ceux qui ont le plus de chance de lui succéder sont mis en gras ou sont surlignés. Ceci a pour objectif d'accroître rapidement l'attention de l'utilisateur vers ces caractères par une mise en saillance. Cependant cette "augmentation" du clavier ne sert que dans le cas des utilisateurs novices. Une fois qu'un utilisateur connaît la disposition des caractères sur le clavier, le temps de recherche d'un caractère est quasiment nul et cet indice visuel n'apporte plus rien.

Une autre utilisation de cette prédiction de caractères consiste à réagencer les caractères en fonction de la saisie de l'utilisateur. Nous avons déjà cité à la section 4.1.1.1, ce type de méthode d'application avec le système LetterWise [MacKenzie 01a] qui réorganise les caractères de chaque touche du clavier téléphonique pour limiter le nombre de frappes à effectuer sur les touches pour saisir les caractères.

Cette technique a aussi été utilisée pour réorganiser l'ensemble des caractères d'un clavier logiciel. Elle a été testée pour des claviers logiciels où les déplacements sur le clavier sont réalisés par défilement d'une touche à l'autre. Ce défilement peut être :

- Soit réalisé automatiquement comme, dans le système Sybille [Schadle 03] (cf. Figure 32) ou encore dans celui de Cantegrit [Cantegrit 01] où l'utilisateur n'a qu'un contacteur pour sélectionner le caractère voulu quand le curseur est sur la touche correspondante ;
- Soit effectué à partir de trois ou cinq boutons servant à se déplacer d'une touche à une autre du clavier. On peut, par exemple, citer le système FOCL[38] [Bellman 98, MacKenzie 02b].

Quel que soit le type de défilement sur ces claviers logiciels, les caractères y sont réagencés après chaque caractère saisi. Le curseur est replacé à un endroit bien précis du clavier (souvent en haut à gauche) et les caractères les plus probables sont placés au plus près pour éviter trop de déplacements (cf Figure 32).



A) B)

FIG. 32 – Le clavier Sybille : B. ré-organisation des caractères après la sélection du caractère 'c' (Figure A.)

Le système SibyLettre [Schadle 02] utilise le modèle statistique n-gram d'ordre 5 pour ses prédictions. Les évaluations théoriques montrent qu'en moyenne la lettre désirée est atteinte en 2,9 défilements avec les prédictions de SibyLettre en défilement linéaire, contre 4,3 pour un défilement ligne/colonne sur un clavier de lettres optimisé avec leur fréquence hors-contexte. Ceci représente un gain de 32% en nombre de défilements et de 50% en nombre de clics nécessaires pour sélectionner une lettre. Une évaluation qualitative [Schadle 04] a également été menée au CMRRF (Centre Mutualiste de Rééducation et Réadaptations Fonctionnelles) de Kerpape auprès de patients IMC (Infirmité Motrice

[38] **F**luctuating **O**ptimal **C**haracter **L**ayout

Cérébrale), montrant une nette préférence pour le système SibyLettre et ce malgré le réarrangement de touches.

Ce type de clavier augmenté semble difficile d'utilisation avec une interaction via un dispositif de pointage (souris, trackball, stylet ...). En effet, si ce système fonctionne relativement bien avec les claviers à défilements, c'est parce que l'utilisateur ne cherche pas à l'avance les caractères sur le clavier : il attend de voir le curseur passé dessus pour le sélectionner. De plus, il est obligé de suivre le défilement imposé, et ne peut pas se déplacer plus rapidement d'un bout à l'autre du clavier. Dans le cas de l'utilisation d'un dispositif de pointage, lorsque l'utilisateur connaît la disposition des touches, il peut anticiper les déplacements vers le caractère suivant. Le réagencement des caractères après une saisie demanderait à l'utilisateur un effort de recherche visuelle supplémentaire et il ne pourrait plus anticiper ses déplacements. Le temps nécessaire pour faire cette recherche visuelle diminuerait alors la vitesse de saisie d'un utilisateur.

Cependant ce réagencement aurait l'avantage de diminuer les déplacements de l'utilisateur, ce qui pour les personnes handicapées des membres supérieurs (qui utilisent un dispositif de pointage) est non négligeable. C'est pourquoi, nous proposons un système qui, sans modifier la disposition des caractères du clavier, propose en complément les caractères les plus probables à proximité.

Nous présenterons dans la suite de cette partie ce système qui se base sur la prédiction de caractères. Nous commencerons par présenter les principes de ce système. Nous détaillerons ensuite les deux premières itérations de notre cycle de conception. Enfin nous conclurons sur ce qu'il est encore possible d'améliorer sur notre système et quelles en sont les perspectives.

7

Le système KeyGlass

Comme démontré dans le chapitre 6, quelle que soit la position des touches et la disposition des caractères sur celles-ci, certains couples de caractères (ou bigrammes) peuvent être éloignés. Pour une meilleure disposition des caractères, les déplacements du dispositif de pointage sont minimisés quand le mot à saisir est composé de bigrammes fréquemment utilisés. Ainsi les deux caractères ont une forte probabilité d'être positionnés à proximité sur le clavier. Au contraire, les bigrammes peu fréquents dans la langue seront éloignés sur le clavier. Les mots qui les utilisent nécessiteront de plus grands mouvements pour les saisir et seront, par conséquent, plus longs à saisir.

7.1 Clavier augmenté par *marking-menu*

7.1.1 Le principe

Isokoski [Isokoski 04b] propose d'augmenter les claviers logiciels[39] en disposant à proximité du caractère saisi un menu proposant un ensemble de caractères (cf. Figure 33). Le menu ajouté a la forme d'un *pie-menu*. Callahan et ses collègues [Callahan 88] montrèrent que ce type d'affichage a l'avantage, par rapport à un menu linéaire, de proposer tous les caractères à distance égale du pointeur du dispositif de pointage. Ceci se traduit par un gain de 15% en vitesse et moins d'erreurs commises avec le *pie-menu*.

Après avoir saisi un caractère sur le clavier, l'utilisateur peut alors saisir un caractère du *pie-menu* sans relever son stylet, en réalisant un simple geste vers le caractère qui l'intéresse sur ce composant supplémentaire. Cette technique d'interaction est connue sous le nom de *marking-menu* [Kurtenbach 93]. Elle a été utilisée par Mackenzie et ses

[39] Cette proposition est faite pour les claviers logiciels utilisés sur dispositifs mobiles et au moyen d'un stylet.

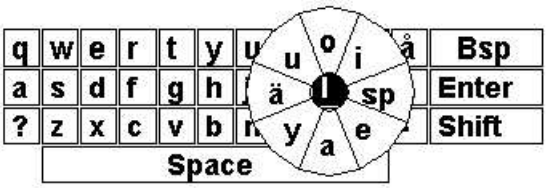


Fig. 33 – Exemple du *marking-menu* sur un clavier logiciel

collègues pour la saisie de caractères numériques [MacKenzie 94, McQueen 94]. Elle offre un double avantage pour l'interaction au stylet :

- A partir du moment où l'utilisateur relève le stylet de l'écran, le *marking-menu* disparaît. Ainsi, si l'utilisateur ne veut pas l'utiliser, ce composant ne le perturbe pas en restant affiché ;
- Lorsqu'un utilisateur connaît parfaitement la position des caractères sur ce *marking-menu*, il peut l'utiliser de manière "aveugle", en effectuant le geste avant même que le *marking-menu* ne soit apparu. Le simple fait de produire le geste valide alors le caractère qui se trouve normalement sur le *marking-menu* à la position pointée par celui-ci.

Après une étude des différentes possibilités d'utilisation, Isokoski a défini un *marking-menu* contenant huit caractères, identiques à toutes les saisies : les six voyelles plus les caractères espace ('**sp**') et retour arrière ('**bs**') (cf Figure 34). Cette configuration a été retenue pour faciliter l'apprentissage des caractères présents sur le *marking-menu* ainsi que leur position sur celui-ci, pour que l'utilisateur puisse rapidement utiliser le *marking-menu* de manière "aveugle".

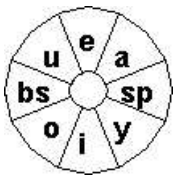


Fig. 34 – *Pie-menu* utilisé dans [Isokoski 04b]

7.1.2 Les points positifs de la technique

La technique du *marking-menu* a l'avantage de rapprocher du dernier caractère saisi un ensemble de caractères. Ceci permet à l'utilisateur de saisir par un simple geste deux caractères, si le second appartient au *marking-menu*. Selon les résultats des expérimentations

menées dans [Isokoski 04b], l'utilisation du *marking-menu* n'améliore la vitesse de saisie d'un utilisateur par rapport à celle du clavier logiciel QWERTY qu'après une vingtaine de sessions d'utilisation ; chaque session durant 15 minutes. Ce temps d'adaptation est expliqué par le fait que les sujets de l'expérimentation sont familiers du clavier QWERTY. Leur conclusion est qu'avec une utilisation régulière du *marking-menu*, les utilisateurs pourraient avoir un gain de temps assez significatif.

L'autre résultat concerne la réduction de la distance. Si cet avantage paraît être minime pour une utilisation du clavier logiciel par une personne valide utilisant un stylet, celui-ci pourrait être apprécié par des personnes handicapées des membres supérieurs qui utilisent des claviers logiciels quotidiennement sur leur ordinateur de bureau.

7.1.3 Considérations pour une meilleure utilisation par des personnes handicapées

La sélection d'un caractère sur le *marking-menu* se fait par un geste au stylet : ce dernier reste appuyé sur l'écran le temps du geste. Ce type d'interaction peut être assez compliqué à réaliser sur un clavier logiciel via un dispositif de pointage telle qu'une souris ou une trackball par une personne ayant des troubles de motricité. En effet, un geste via un stylet sur l'écran correspond à une action de déplacement de la souris couplée à la pression d'un bouton (activité motrice difficile et engendrant des effets de fatigue).

Par contre, l'avantage majeur de ce système pour des personnes handicapées est la minimisation des distances à parcourir. Le fait de prendre un *marking-menu* avec huit caractères fixes diminue la probabilité d'avoir le caractère adéquat sur le *marking-menu* : 35% des caractères peuvent être saisis via le *marking-menu* représenté à la Figure 34. Il n'y a pas de récursivité dans le système proposé par [Isokoski 04b] : en effet, après qu'un caractère ait été saisi avec le *marking-menu*, un nouveau n'apparaît pas. Il faut obligatoirement saisir le caractère suivant sur le clavier. Ce procédé réduit encore la possibilité de diminuer les distances.

De cette analyse des limites de la technique à base de *marking-menu*, nous proposons un système basé sur ce principe de rapprochement de caractères, mais avec le souci *a priori* d'une utilisabilité accrue par des personnes handicapées des membres supérieurs.

7.2 Le système KeyGlass

7.2.1 Le principe

Afin de répondre à ce besoin, nous avons conçu un système qui propose à l'utilisateur des caractères supplémentaires au fur et à mesure de la saisie associés à des touches ajoutées dynamiquement à proximité du dernier caractère saisi. Nous nommerons ce système KeyGlass. A la différence de ce qui a été réalisé par Isokoski, nous avons fait le choix de proposer après chaque caractère saisi les caractères qui ont le plus de probabilités de lui succéder. Même si le temps de recherche visuelle du caractère pourrait être plus long, nous souhaitons proposer le plus souvent possible le bon caractère à proximité du dernier saisi, et ainsi réduire au maximum les distances de saisie. Pour augmenter les probabilités de diminuer les distances, le système fonctionne aussi par récurrence : après chaque nouvelle saisie d'un caractère, que ce soit sur une touche fixe ou une KeyGlass, de nouveaux caractères sont proposés autour de ce dernier.

A l'instar des *toolglasses* [Bier 93] qui proposent une palette semi-transparente d'outils à proximité du pointeur, nous avons choisi d'afficher les caractères supplémentaires sur des touches rondes semi-transparentes (cf Figure 35). Nous avons préféré retenir la semi-transparence pour donner à l'utilisateur une vision globale du clavier et ainsi ne jamais masquer un caractère par l'affichage d'un autre qui viendrait se positionner par dessus. La sélection des caractères se fait de la même manière que ce soit un caractère associé à une touche du clavier fixe ou à une touche KeyGlass par un clic sur la dite touche.

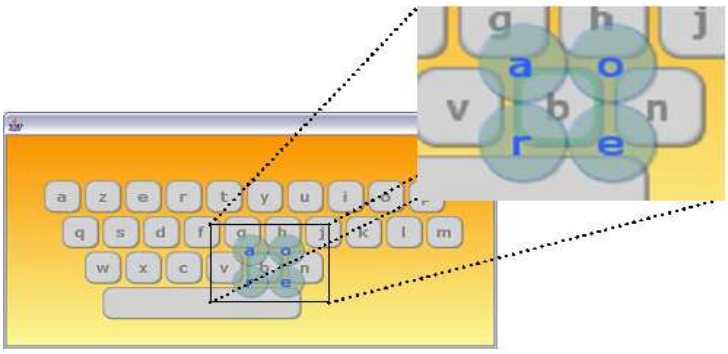


Fig. 35 – Quatre KeyGlasses positionnées après la saisie du caractère 'b'

7.3 Architecture du système de saisie

7.3.1 Un système modulaire

Nous avons conçu le système de manière modulaire pour pouvoir changer une partie du fonctionnement sans avoir à changer le reste. Notre système est consitué de trois parties (voir Figure 36) :

- **La partie fixe du clavier logiciel :** elle concerne l'ensemble des touches qui ne bougeront pas pendant la saisie. Les caractères associés à ces touches sont aussi fixes. Cet ensemble de touches/caractères est décrit dans un format XML (cf. Annexe B, section B.9). Nous avons retenu un langage XML car il permet de modifier facilement les caractéristiques du clavier de base. Nous pouvons ainsi changer la forme, la position, les couleurs d'une touche ainsi que le caractère qui y est associé. Cette partie fixe est par conséquent complètement indépendante des KeyGlasses qui peuvent être ajoutées par dessus ;
- **Le système de prédiction :** il reçoit les caractères saisis par l'utilisateur. A partir de ces derniers, il renvoie au gestionnaire d'affichage des KeyGlasses un classement des caractères qui ont le plus de probabilité de succéder à celui qui vient d'être saisi ;
- **Le gestionnaire d'affichage des KeyGlasses :** il gère l'affichage des KeyGlasses au dessus du clavier fixe. Il a à sa charge : la position des KeyGlasses, leur forme, leur couleur. Il leur associe le caractère en fonction de ce qu'il reçoit du système de prédiction.

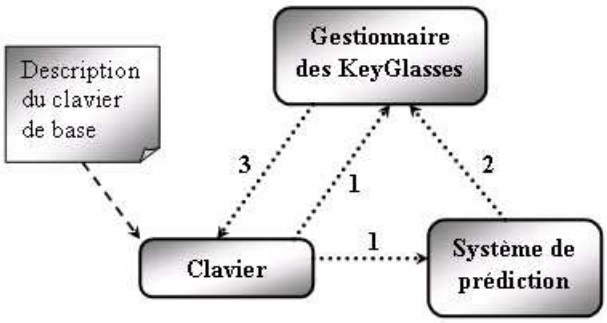


FIG. 36 – Architecture du système KeyGlass

7.3.2 Mode de fonctionnement

La figure 36 montre le principe de fonctionnement du système KeyGlass.

1. Lorsqu'un caractère est saisi sur le clavier logiciel, celui-ci est envoyé au système de prédiction. Dans le même temps, les coordonnées du pointeur au moment de la sélection du caractère[40] sont envoyées au gestionnaire de KeyGlasses ;
2. Le système de prédiction calcule la probabilité de chaque caractère de succéder au dernier caractère reçu, puis les classe par ordre croissant en fonction de leur probabilité d'apparaître. Il renvoie alors ce classement au gestionnaire d'affichage de KeyGlasses ;
3. Enfin le gestionnaire d'affichage des KeyGlasses, après avoir déterminé la position des futures KeyGlasses, affecte à chacune d'elles un caractère et renvoie toutes ces informations au clavier logiciel pour que celui-ci les affiche.

Notons que les KeyGlasses une fois affichées au dessus du clavier sont considérées par le système comme appartenant au clavier logiciel. Ainsi comme nous l'avons mentionné dans la section 7.2.1, ce processus peut fonctionner de manière récursive. L'appui sur une KeyGlass pour saisir un caractère entraîne exactement le même processus que celui que nous venons de décrire ci-dessus.

7.3.3 Protocole de communication inter-module

De la même manière que pour la plate-forme E-Assiste, nous avons utilisé le bus logiciel Ivy [Buisson 02] pour la communication entre les différents modules. Pour faciliter le remplacement d'un module par un autre, nous avons défini un protocole de communication entre les modules (voir Annexe A). Ainsi, chaque module connait exactement ce qu'il peut recevoir et/ou envoyer. Le remplacement d'un module par un autre est facile : il suffit que ce dernier soit connecté sur le bus Ivy et respecte le protocole de communication.

Afin d'être compatible avec la plate-forme E-Assiste, ce protocole de communication reprend les règles de celui de E-Assiste (voir Annexe A). Ainsi les divers messages respectant ce protocole peuvent être aussi collectés par la plate-forme E-Assiste. Ceci nous permettra d'étudier rapidement l'utilisabilité du système KeyGlass à partir d'expérimentations menées sur la plate forme E-Assiste (voir Chapitre 8).

[40] Cette sélection peut être caractérisée par exemple par la pression d'un bouton du dispositif de pointage.

7.4 Le système de prédiction de caractères utilisé

7.4.1 Description des systèmes testés

Nous avons choisi d'étudier plus particulièrement deux systèmes de prédiction :

- **Un système à base de bigrammes,** qui propose pour un caractère donné, les caractères qui ont le plus de chance de lui succéder. Ce procédé ne prend en compte pour la prédiction que le caractère qui vient d'être saisi. On retrouve ainsi toujours, pour un caractère donné, les mêmes caractères prédits. Ces caractères prédits ont l'avantage de pouvoir être appris par une personne qui les utiliserait régulièrement. Notre table de bigrammes a été déduite de la base de données BDLEX [De Calmès 98] ;
- **Un système à base d'arbre lexicographique,** construit à partir d'un ensemble de mots (cf. Figure 37). Chaque mot est représenté par un chemin de la racine à une feuille de l'arbre, où chaque caractère est un nœud de l'arbre. Deux mots ayant le même préfixe utilisent le même chemin initial pour éviter de créer trop de nouveaux nœuds. Pour les mots qui sont préfixes d'autres mots, une feuille spéciale 'fin de mot' (cercle grisé sur la Figure 37) a été créée. Comme [Ménier 01], nous avons choisi de donner un poids à chaque arête de l'arbre. A chaque mot ajouté, les arêtes entre les différents nœuds traversés sont incrémentées de 1. Ceci permet de donner une importance à chaque nœud. Notre arbre lexicographique a été construit à partir des 140 000 mots les plus fréquents de la langue française.

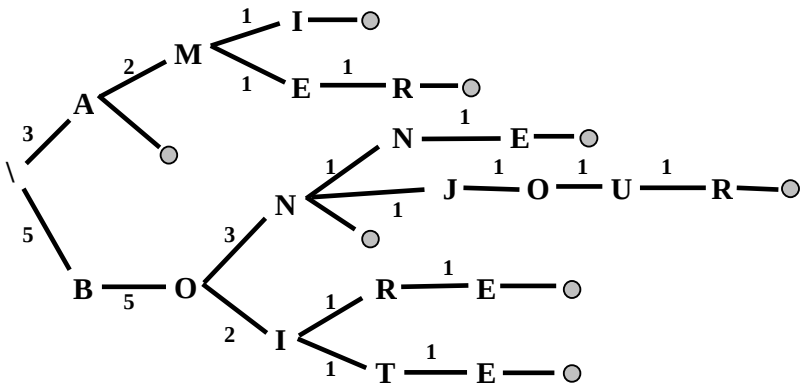


FIG. 37 – Arbre lexicographique construit à partir des mots a, ami, amer, bon, bonne, bonjour, boire, boite.

Ce système prend en compte le début du mot qui vient d'être saisi en parcourant l'arbre en fonction de ce qui a déjà été saisi. Par rapport à ce début de mot, il classe les caractères qui peuvent lui succéder en fonction de leur probabilité d'apparition

(plus la valeur de l'arête est grande, plus le caractère a de chance d'apparaître). L'avantage de ce système est qu'il propose, après le début de mot déjà saisi, que les caractères qui sont susceptibles de former un mot connu. L'inconvénient est que si l'arbre lexicographique ne contient pas un mot dans son corpus d'apprentissage, ou si l'utilisateur fait une faute lors de la saisie de son mot sans la corriger, la prédiction risque d'être fausse par rapport à ce que veut saisir l'utilisateur.

7.4.2 Evaluation de ces systèmes

Afin de déterminer le système de prédiction le plus performant à intégrer dans notre système KeyGlass, nous avons fait une simulation par ordinateur pour évaluer les performances de notre système avec chacun d'eux. Celle-ci consistait à simuler la saisie d'un ensemble de mots. Nous avons ainsi réalisé 3 sessions de simulation : une avec un clavier Azerty (limité aux 26 caractères de l'alphabet et l'espace), une avec le système KeyGlass et le système de prédiction à base de bigrammes et une autre avec le système KeyGlass et le système de prédiction à base d'arbre lexicographique.

Chaque session de simulation a consisté à saisir l'ensemble des mots de l'Officiel du Scrabble (ODS). Ce dictionnaire comporte 364 370 mots, soit 4 005 510 caractères[41]. Nous avons simulé la saisie avec l'utilisation de quatre KeyGlasses. Pour déterminer la pertinence des deux systèmes de prédiction, nous avons étudié plus particulièrement le taux de caractères correctement prédits : c'est-à-dire le nombre de caractères qui peuvent être saisis directement à partir d'une KeyGlass.

7.4.3 Analyse des résultats

Le tableau 9 présente les résultats de ces simulations où :

- *Nb caractères* représente le nombre de caractères correctement prédits par le système et affiché sur une KeyGlass, parmi les 4 005 510 à saisir ;
- *Tx prédiction* donne les performances du système de prédiction en taux de bonne prédiction.

Comme nous pouvons le constater sur le Tableau 9, les performances *a priori* seraient meilleures avec un système de prédiction basé sur un arbre lexicographique. Cependant, une étude plus minutieuse de ces résultats montre que ces deux systèmes sont assez complémentaires. Pour cela, nous avons réalisé un décompte des caractères correctement prédits après chaque caractère saisi dans le mot et ce, pour chaque mot. Cette étude montre

[41] Le nombre de caractères a été calculé en prenant pour chaque mot la taille de la chaîne de caractères plus un caractère espace à la fin.

Type de prédiction	*Nb caractères*	*Tx prédiction*
bigramme	**2 449 235**	*61,15*
arbre lexico	**2 499 692**	*62,40*

TAB. 9 – Résultat des simulations avec différents systèmes de prédiction

une progression différente pour les deux systèmes. L'évolution des performances des deux systèmes est présentée sous forme d'histogramme sur la figure 38. L'axe des abscisses représente le nombre de caractères du mot qui ont été saisis au moment de la prédiction. L'axe des ordonnées donne le pourcentage de caractères correctement prédits par rapport au nombre de caractères à saisir. Nous voyons ainsi qu'un système de prédiction de type arbre lexicographique a de meilleures performances que celui basé sur les bigrammes sur les six ou sept premières lettres. Par contre, pour les mots longs, le système de prédiction à base de bigrammes est meilleur sur les dernières lettres du mot que l'arbre lexicographique. Ceci peut s'expliquer par le fait que l'arbre lexicographique a été construit à partir des 140 000 mots les plus fréquents de la langue française qui ne comportent pas l'ensemble des formes fléchies que l'on retrouve dans l'Officiel du Scrabble. Les formes fléchies étant souvent les mêmes, on obtient une fréquence élevée de ces bigrammes. Ainsi, le système de prédiction bigramme prend en compte ces fréquences élevées de bigrammes.

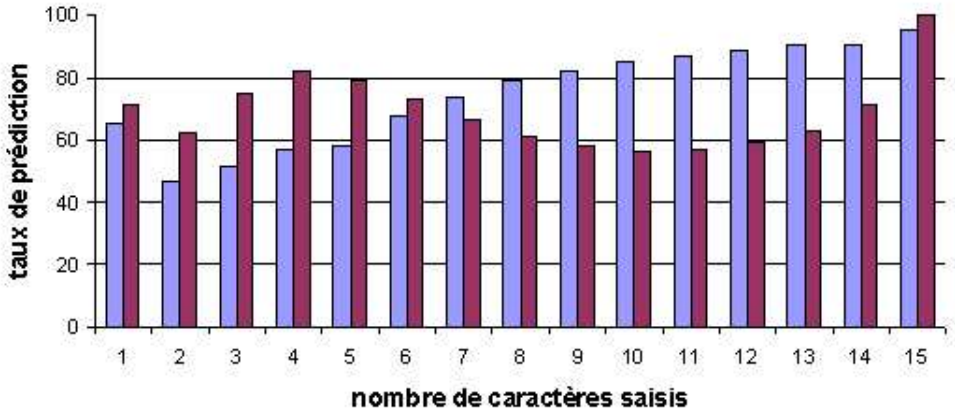


FIG. 38 – Taux de prédiction correcte en fonction du système de prédiction et de l'avancée dans le mot

Les avantages respectifs de ces deux systèmes nous ont amené à évaluer le gain que nous apporterait une combinaison des deux.

7.4.4 Le système de prédiction final

A partir de cette analyse, nous avons modélisé un système de prédiction prenant en compte à la fois l'arbre lexicographique et le bigramme. Pour chaque prédiction de caractères, le système prend les résultats en sortie de ces deux systèmes et les pondère respectivement par une variable P_a pour l'arbre lexicographique et P_b pour le bigramme. Ces deux valeurs sont comprises entre 0 et 1 et $P_a = 1 - P_b$.

Ainsi pour connaître les meilleures valeurs à attribuer à P_a et P_b, nous avons réalisé une série de simulations où nous faisons varier le poids des deux systèmes de prédiction. Nous avons commencé avec $P_a = 0,95$ et $P_b = 0,05$ puis nous les avons respectivement décrémenté et incrémenté par pas de 0.05 jusqu'à obtenir les valeurs inverses du départ ($P_a = 0,05$ et $P_b = 0,95$).

P_a	P_b	P	P_a	P_b	P
0,95	0,05	74,83 %	0,45	0,55	72,90 %
0,90	0,10	74,86 %	0,40	0,60	72,52 %
0,85	0,15	74,69 %	0,35	0,65	72,07 %
0,80	0,20	74,53 %	0,30	0,70	71,65 %
0,75	0,25	74,36 %	0,25	0,75	71,01 %
0,70	0,30	74,23 %	0,20	0,80	70,42 %
0,65	0,35	73,94 %	0,15	0,85	69,58 %
0,60	0,40	73,73 %	0,10	0,90	68,48 %
0,55	0,45	73,52 %	0,05	0,95	65,78 %
0,50	0,50	73,23 %			

Tab. 10 – Performance du système de prédiction en fonction de la pondération de l'arbre lexicographique et du bigramme

Le tableau 10 présente les résultats obtenus pour l'ensemble des pondérations testées. La meilleure solution consiste à prendre en compte les prédictions de l'arbre lexicographique à 90% et celle du bigramme à 10%. Cette combinaison a été retenue pour être le système de prédiction de base du système KeyGlass.

7.4.5 Le nombre de caractères proposés

Afin de savoir quel nombre de KeyGlasses il est intéressant d'afficher après chaque caractère, nous avons réalisé une dernière simulation. Celle-ci consistait aussi à simuler la saisie de l'ensemble des mots de l'Officiel du Scrabble. Nous avons réalisé 12 sessions : nous sommes partis d'une simulation avec une seule KeyGlass (un seul caractère prédit), et nous en avons rajouté une de plus par session, jusqu'à en avoir 12.

Nb	Prediction		Fixe	
1	1 422 439	35,51 %	500 675	12,50 %
2	2 296 198	57,33 %	844 699	21,09 %
3	2 704 554	67,52 %	1 209 069	30,19 %
4	2 994 645	74,76 %	1 534 980	38,32 %
5	3 195 508	79,78 %	1 864 918	46,56 %
6	3 299 010	82,36 %	2 138 823	53,40 %
7	3 384 816	84,50 %	2 397 716	59,86 %
8	3 441 196	85,91 %	2 629 438	65,65 %
9	3 484 235	86,99 %	2 840 360	70,91 %
10	3 522 484	87,94 %	2 979 008	74,37 %
11	3 555 267	88,76 %	3 111 639	77,68 %
12	3 574 288	89,23 %	3 199 115	79,87 %

TAB. 11 – Performance du système de prédiction en fonction du nombre de KeyGlass disponibles

Nous avons pris pour cette simulation le système de prédiction optimal précédemment calculé (c'est-à-dire le système couplant arbre lexicographique et bigramme en les pondérant respectivant par 0,9 et 0,1). A titre de comparaison, nous avons également réalisé la même série de simulations en proposant toujours les mêmes caractères (nous avons pris les caractères les plus fréquemment utilisés pour la langue française).

Le tableau 11 présente l'ensemble de ces résultats. Nous avons choisi pour la suite de notre conception de retenir quatre KeyGlasses. Ce choix s'explique par le fait que nous obtenons un taux d'utilisation des KeyGlasses de 74,76% avec 4 caractères prédits, ce qui est à peu près équivalent au taux que l'on obtiendrait avec 10 caractères fixes, soit plus que ce qui est proposé par Isokoski [Isokoski 04b]. De plus, ceci nous paraît être un bon compromis entre le taux de prédiction (près de trois caractères sur quatre sont correctement prédits, ce qui équivaut à la présentation de dix caractères fixes) et l'affichage de celle-ci à l'écran (cela permet d'en mettre une à chaque coin de touche par exemple).

Pour la conception de notre système de saisie, nous avons fait le choix d'afficher quatre KeyGlasses et de proposer les caractères qui y sont associés à partir d'un système de prédiction hybride entre un arbre lexicographique et une table bigramme. Nous allons maintenant étudier et évaluer par des expérimentations avec des sujets valides et des sujets handicapées des membres supérieurs l'utilisabilité de ces quatre KeyGlasses.

8

Méthode de conception centrée utilisateur

Ce chapitre présente maintenant le cycle de conception utilisé pour aboutir au système KeyGlass actuel. Nous avons procédé à une démarche de conception itérative centrée utilisateur. Nous avons mené deux itérations complètes que nous allons maintenant vous présenter.

8.1 Première itération

8.1.1 Description du principe d'affichage des KeyGlasses

Dans la première version du clavier optimisé par KeyGlass, nous avons cherché à minimiser les distances à parcourir. Le gestionnaire de KeyGlasses positionne les quatre KeyGlasses autour du pointeur du dispositif de pointage. La position des KeyGlasses est calculée en fonction des coordonnées du pointeur au moment de la pression sur une touche normale ou une KeyGlass du clavier (voir Figure 39). A partir de ces coordonnées, quatre zones sont formées pour positionner les KeyGlasses (Figure 39). Les Keyglasses sont placées de manière à coller aux axes. Cette manière de positionner les KeyGlasses a l'avantage d'être au plus près du pointeur sans qu'il n'y ait de recouvrement entre les KeyGlasses.

Pour cette première itération, nous avons considéré le clavier AZERTY comprenant les 26 caractères de l'alphabet romain ainsi que le caractère espace. Le système de prédiction est celui retenu suite aux différentes simulations de la section 7.4. Il ne repose que sur la prédiction de ces 27 caractères. La figure 40 montre un exemple du clavier utilisé pour

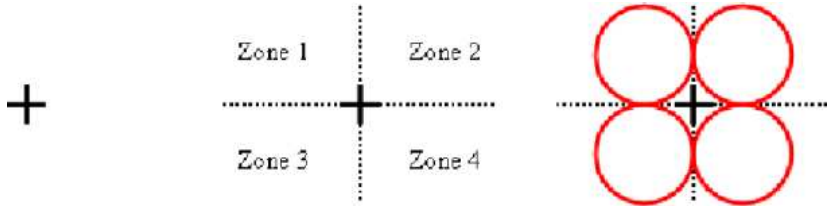


FIG. 39 – Designation des positions des KeyGlass

cette première itération ainsi que les KeyGlasses qui ont été positionnées après la saisie du caractère 'b'.

8.1.2 Evaluation théorique

Cette évaluation théorique a pour objectif de calculer le gain que pourrait apporter l'utilisation du système à un utilisateur. Rappelons les deux hypothèses faites pour ce système : d'une part, diminuer les distances à parcourir avec le dispositif de pointage, et d'autre part, augmenter la vitesse de saisie de texte.

Ce sont ces deux variables (distance et vitesse) que nous avons cherchées à quantifier par cette simulation. A des fins de comparaisons, nous avons réalisé une simulation en utilisant le clavier de base sans utilisation de KeyGlasses, puis la même simulation avec les KeyGlasses.

Pour ce système, l'évaluation théorique conduite pour l'évaluation du clavier GAG (basée sur la formule de Soukoreff, cf. section 1.3.2) n'est pas possible. En effet, l'aspect dynamique du système ne permet pas de prévoir toutes les situations possibles comme

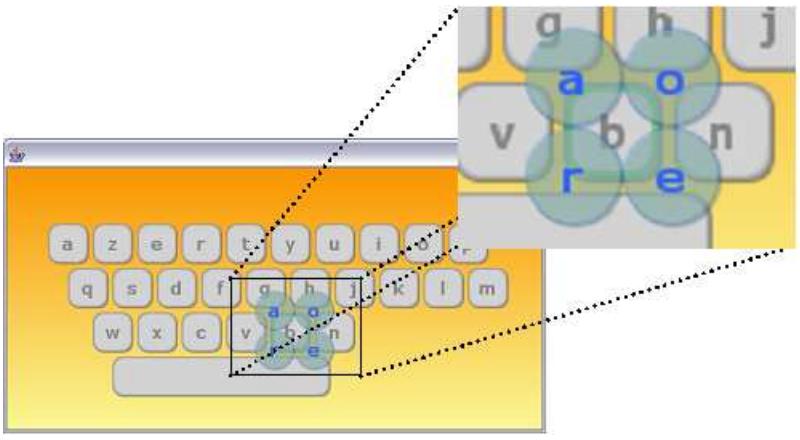


FIG. 40 – Quatre KeyGlasses positionnées après la saisie du caractère 'b'

sur un clavier fixe. La saisie de chaque mot est unique : nous avons, comme pour les précédentes, réalisé l'évaluation en simulant la saisie de l'ensemble des mots de l'Officiel du Scrabble.

Pour calculer les distances de chaque mot, nous avons pris à chaque fois comme coordonnées de pression d'une touche, le centre de la touche concernée. Ce choix est justifié par l'étude de Zhai et ses collègues [Zhai 02b] qui a montré que lors de la sélection d'une touche, la répartition des impacts d'un stylet sur celle-ci était concentrée autour du centre de la touche. La distance d'un mot est alors obtenue en additionnant la longueur de chaque segment séparant les centres de deux touches qui viennent d'être pressées successivement. Cette simulation de saisie de texte est dite "parfaite" :

1. la saisie de chaque mot est effectuée sans erreur de frappe ;
2. dans le cas de la simulation avec le système KeyGlass, si le caractère à saisir est prédit et apparaît sur une KeyGlass, la simulation choisit à chaque fois le caractère se trouvant sur la KeyGlass. Sinon il le prend sur la disposition de touches du clavier de base.

Pour le calcul du temps, on distingue généralement deux classes d'utilisateurs : les novices et les experts (cf. section 1.3.2). Les performances obtenues par ces deux types d'individus sont généralement considérées comme les bornes des performances que l'on peut espérer avoir avec le système en question. Notre système étant composé de deux parties (un clavier fixe et des touches supplémentaires déplacées dynamiquement au cours de la saisie), nous avons choisi d'étudier trois cas :

1. L'utilisateur est novice à la fois avec le clavier fixe et les KeyGlasses. En plus du temps de déplacement, nous comptons pour chaque caractère à saisir, un temps de recherche visuelle de ce dernier parmi les 31 possibilités (les 27 caractères du clavier fixe plus les quatre KeyGlasses) ;
2. L'utilisateur est expert sur le clavier de base, mais novice avec les KeyGlasses : pour ce type d'utilisateur, le temps de recherche visuelle se limite à la recherche parmi les quatre KeyGlasses (la recherche visuelle est considérée comme nulle sur le clavier fixe, vu que l'on considère l'utilisateur comme expert sur ce support) ;
3. L'utilisateur est expert avec l'ensemble du système : le temps de recherche est considéré comme négligeable. Le temps pour saisir un mot se limite au temps de déplacement d'un caractère à un autre.

L'estimation du temps pour se déplacer est prédite par la loi de Fitts (cf section 1.2.1) alors que celle du temps de recherche visuelle d'une touche est formulé par la loi de Hick-Hyman (cf section 1.1). Nous avons choisi de prendre comme valeur pour ces différences lois :

- $a = 0, b = 4,5$, et $MT_{repeat} = 0,127$ pour la loi de Fitts. Nous avons repris les mêmes valeurs calculées pour le pointage de cible avec une souris pour la loi de Fitts (cf. Tableau 1).
- $a = 0$ et $b = 0,2$ pour la loi de Hick-Hyman ; ce sont les valeurs utilisées pour la loi de Hick-Hyman pour prévoir le temps maximum qu'un utilisateur va passer à rechercher un caractère (cf. section 1.3.3.2).

Ainsi, le temps nécessaire pour saisir un mot est la somme des temps de déplacement et de recherche visuelle pour chaque caractère saisi.

Type de clavier		AZERTY	*KeyGlass*	*Différence*
Distance *en pixels*		**1206,19**	**483,15**	*-59,94 %*
Temps	Novice	**10776 ms**	**10802 ms**	*0,24 %*
	Expert / Novice	**3270 ms**	**5493 ms**	*67,98 %*
	Expert	**3270 ms**	**1860 ms**	*-43,12 %*

TAB. 12 – Résultat en distance et temps des simulations selon les classes d'utilisateur

Le tableau 12 présente les résultats de ces simulations. La colonne AZERTY donne les résultats pour le clavier fixe sans utilisation de KeyGlasses, la colonne *KeyGlass* représente ceux de ce même clavier avec utilisation du système KeyGlass. Enfin la colonne *Différence* donne en pourcentage la différence qui existe entre une utilisation du clavier fixe sans et avec KeyGlasses. Enfin les distances sont exprimées en nombre de pixels par mot, et le temps représente le temps moyen en millisecondes pour saisir un mot.

Rappelons que le taux de prédiction de notre système de prédiction est de l'ordre de 75% dans cette configuration. Nous constatons que si l'utilisateur saisit, dès que cela est possible, les caractères positionnés sur les KeyGlasses, il peut réduire la distance à parcourir d'environ 60%. Par contre, les gains en vitesse de saisie sont moins importants. Si un utilisateur connaît la disposition touches/caractères du clavier sur lequel il va utiliser pour la première fois les KeyGlasses, le gain en temps est négatif (-67,98 %). Par contre, si l'utilisateur ne connaît pas la disposition des caractères du clavier sur lequel il va saisir, les temps de saisie seront alors quasi identiques entre le clavier avec ou sans le

système KeyGlass (en revanche, l'utilisateur y gagnera en distance). A partir du moment où l'utilisateur est en situation d'expert (c'est-à-dire après plusieurs heures d'utilisation) avec le clavier et le système KeyGlass, l'apport de ce dernier peut alors devenir intéressant même en terme de vitesse : on peut ainsi espérer une diminution du temps de l'ordre de 40% dans le meilleur des cas.

8.1.3 Evaluation

A la suite de cette simulation, nous avons souhaité réaliser une première expérimentation avec des sujets pour d'une part, vérifier nos simulations, et d'autre part, connaître leurs impressions sur le système : savoir, par exemple, si la manière de présenter les KeyGlasses leur convenait ou non. L'expérimentation est réalisée avec le système décrit précédemment. Le clavier logiciel de base est un clavier AZERTY. Ce clavier étant utilisé par un grand nombre de français, il était difficile de disposer de sujets novices avec ce clavier. Nous nous situons par conséquent dans le cas où les sujets sont experts (ou du moins non novices) avec le clavier de base mais novices avec les KeyGlass. L'hypothèse que nous souhaitions vérifier lors de cette expérimentation est : y a-t-il oui ou non une diminution de la distance parcourue lors de l'utilisation du système KeyGlass ?

8.1.3.1 Protocole expérimental

Profil des sujets

Bien que ce système soit prévu au départ pour une utilisation par des personnes handicapées des membres supérieurs, nous avons réalisé cette première expérimentation avec une majorité de personnes valides. Nous avons fait ce choix puisqu'il est assez difficile de rassembler un nombre suffisant de personnes handicapées pour réaliser une expérimentation. Cette première expérimentation avait aussi pour but d'identifier certains problèmes d'interaction. Les remarques des sujets valides avaient aussi pour objectif de permettre d'améliorer l'utilisabilité de notre système dans une deuxième itération de la conception de ce système.

L'expérimentation a été menée auprès de 30 sujets valides et d'un sujet handicapé. La plupart des sujets sont des informaticiens et utilisent le clavier AZERTY au quotidien.

Déroulement de l'expérimentation

L'expérimentation s'est déroulée en quatre phases :

- **une phase d'entraînement,** où les sujets devaient saisir une douzaine de mots à l'aide de notre système KeyGlass ;

- **trois exercices de recopie :**
  - un exercice avec le clavier logiciel de base ;
  - un exercice avec le clavier logiciel de base augmenté du système KeyGlass ;
  - le même exercice que le précédent, mais avec un délai d'affichage pour les KeyGlasses ; si au bout d'une seconde et demie aucun caractère n'avait été saisi, les KeyGlasses disparaissaient. Nous avons réalisé cet exercice supplémentaire pour déterminer si l'affichage de KeyGlass était un facteur gênant pour l'utilisateur quand le caractère à saisir n'était pas sur les KeyGlass mais sur le clavier fixe.

Chaque exercice consistait à saisir un ensemble de 30 mots. Cet ensemble de mots était le même pour chaque exercice et chaque sujet. La liste de ces mots est donnée dans le tableau 3 de la section 2.4.

	Exercice 1	Exercice 2	Exercice 3
Groupe 1	AZERTY	KeyGlass	KeyGlass *(délai)*
Groupe 2	AZERTY	KeyGlass *(délai)*	KeyGlass
Groupe 3	KeyGlass *(délai)*	AZERTY	KeyGlass
Groupe 4	KeyGlass *(délai)*	KeyGlass	AZERTY
Groupe 5	KeyGlass	AZERTY	KeyGlass *(délai)*
Groupe 6	KeyGlass	KeyGlass *(délai)*	AZERTY

TAB. 13 – Ordre d'éxecution des exercices en fonction du groupe

Pour éviter les effets d'apprentissage des mots, nous avons contre balancé l'ordre d'exécution des trois exercices. Pour cela nous avons constitué 6 groupes de 5 sujets valides. Chaque groupe réalisait les exercices dans un ordre différent (cf. Tableau 13). L'expérimentation a eu lieu au sein de notre laboratoire des usages et fut réalisée au moyen de la plate-forme E-ASSISTE.

8.1.3.2 Analyse des résultats

Résultats des sujets valides

Les Figures 41 et 42 présentent respectivement les résultats en distance et en vitesse de l'expérimentation menée par les 30 sujets. Les bâtons bleus, verts et jaunes représentent respectivement les performances obtenues avec le clavier AZERTY seul, celles du clavier

AZERTY avec les KeyGlasses, sans et avec délai sur le temps d'affichage. Sur chaque graphique, nous présentons les performances moyennes de chaque groupe ainsi que la performance moyenne de la globalité des sujets. Les distances sont mesurées en nombre de pixels parcourus pour saisir l'ensemble des mots. Le temps correspond au nombre de secondes qui ont été nécessaires à l'utilisateur pour saisir l'ensemble des mots. Les mesures de temps pour chaque mot sont prises entre la saisie du premier et du dernier caractère du mot.

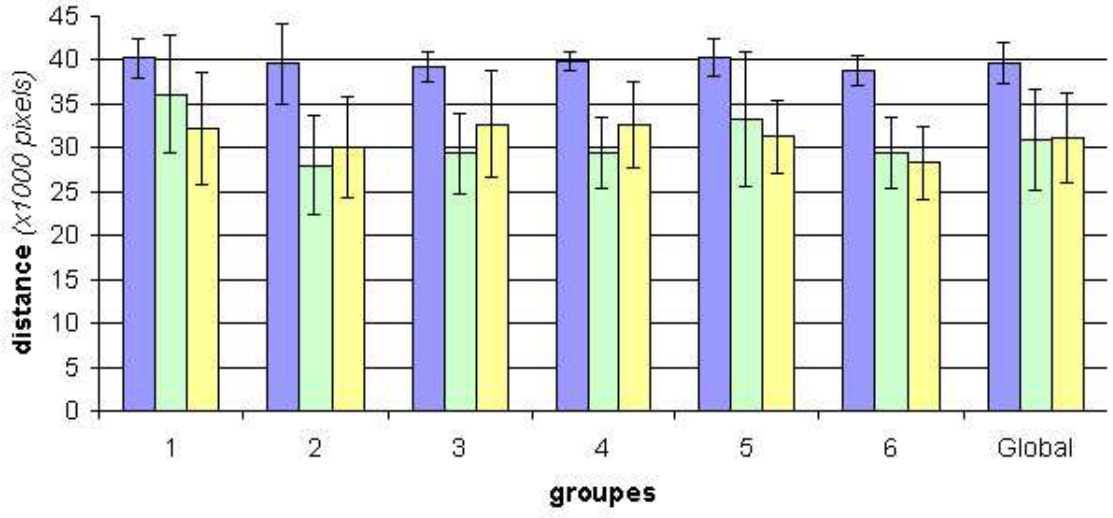


FIG. 41 – Distances nécessaire en moyenne pour réaliser la tâche de saisie

Nous constatons que l'ordre dans lequel les exercices ont été réalisés n'a eu que peu d'impact sur les résultats entre le clavier fixe seul et le clavier avec le système KeyGlass. Par conséquent, nous présenterons principalement notre analyse sur les résultats globaux de l'expérimentation.

Sur l'ensemble des 30 sujets, la distance parcourue pour saisir les 30 mots avec le clavier AZERTY augmenté du système KeyGlass a été réduite de près de 25% par rapport au même exercice avec seulement le clavier AZERTY.

Sur l'histogramme représentant les temps, il apparaît que les utilisateurs ont été bien plus rapides pour saisir les 30 mots avec le clavier AZERTY seul qu'avec le système KeyGlass en plus. En moyenne sur les 30 sujets, nous obtenons un temps moyen de 133 secondes avec AZERTY seul contre 223 secondes avec le système KeyGlass en plus. Ceci représente une augmentation de 67,67%. Nous constatons aussi une légère incidence sur l'ordre d'exécution des deux exercices avec les KeyGlass : le second exercice est toujours meilleur que le premier. Cette différence est d'environ 10,5% en moyenne sur les six groupes

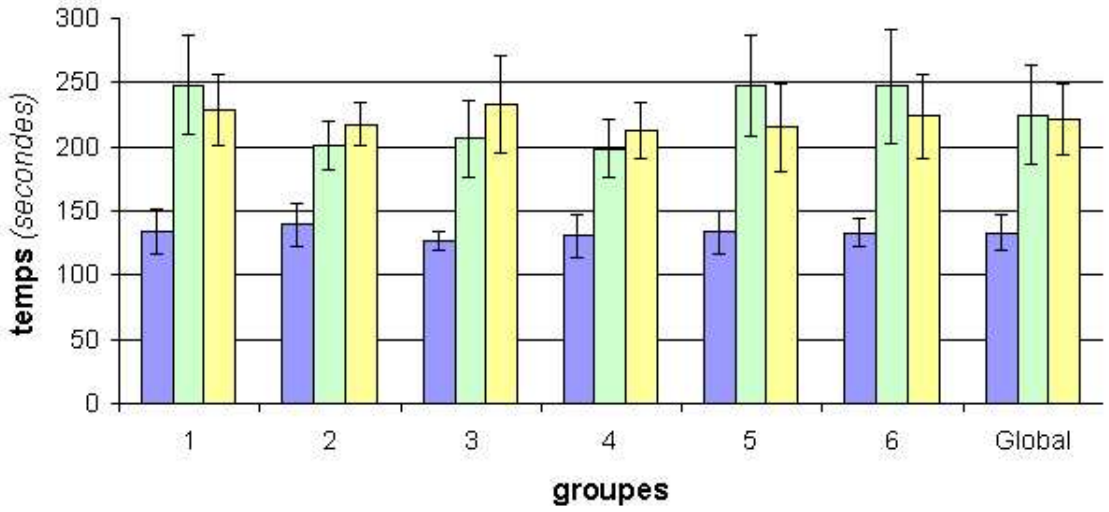


FIG. 42 – Temps nécéssaire en moyenne pour réaliser la tache de saisie

entre le premier exercice avec KeyGlass et le second. On peut attribuer cette différence au fait que les sujets, entre le premier et le second exercice, retiennent la position de certains caractères sur les KeyGlasses et mettent moins de temps à chercher les caractères lors du second exercice.

Résultats du sujet handicapé

Ce sujet a réalisé les exercices dans l'ordre suivant : KeyGlass avec délai, KeyGlass sans délai, AZERTY seul. Le tableau 14 présente ses résultats. On peut noter que ces derniers sont du même type que ceux des sujets valides : les distances sont une nouvelle fois minimisées, néanmoins le temps de saisie reste encore supérieur avec le système KeyGlass.

Cependant, notons que les temps de saisie sont supérieurs à ceux d'un sujet valide, avec respectivement 241 secondes et 314 secondes (moyenne des deux exercices de recopie avec le système KeyGlass) pour les exercices AZERTY et KeyGlass. Néanmoins, il est à souligner que la différence entre les deux types d'exercice est aussi bien moins importante (environ 30% de temps en plus avec le système KeyGlass pour la personne handicapée contre plus de 65% pour les personnes valides).

8.1.3.3 Discussion

On peut dire après cette première expérimentation que l'hypothèse sur la diminution de la distance est bien vérifiée, que ce soit par une utilisation par des personnes valides ou handicapées des membres supérieurs.

Type de clavier	Distance	Temps
AZERTY	50658	241
AZERTY + KeyGlass *(avec délai)*	45851	322
AZERTY + KeyGlass *(sans délai)*	41711	306

TAB. 14 – Résultat de l'expérimentation avec un sujet handicapé

Comme l'ensemble des sujets qui ont fait cette expérimentation connaissaient bien le clavier AZERTY, les résultats obtenus pour le temps de saisie sont assez conformes aux simulations que nous avions réalisées auparavant. Ils étaient experts avec le clavier de base et novices avec les KeyGlasses. On peut par conséquent attribuer une partie de ce temps supplémentaire à la recherche des caractères sur les KeyGlasses alors que lors de la saisie sans KeyGlass, les sujets ont eu tendance à anticiper le déplacement.

8.1.4 Retours utilisateur

Trois remarques principales sont ressorties des échanges que nous avons pus avoir avec les sujets à la fin de l'expérimentation :

- Le fait que les KeyGlasses disparaissent automatiquement après la saisie d'un caractère ; ceci pose un problème lorsque l'on souhaite saisir deux fois consécutivement un même caractère à partir d'une KeyGlass. Après la saisie d'un caractère sur une KeyGlass, celle-ci disparaît et l'utilisateur ne peut pas saisir une seconde fois son caractère sur la KeyGlass. Ceci oblige l'utilisateur à rechercher à nouveau le caractère parmi les nouvelles KeyGlasses ou sur le clavier ; cette recherche lui fait perdre *a priori* beaucoup plus de temps que lorsqu'il peut effectuer un double clic pour saisir une double lettre ;
- Certains sujets ont trouvé déroutant le fait que les KeyGlasses se déplacent tout le temps, et ce, même pour suivre le pointeur du dispositif de pointage. Ils auraient préféré des KeyGlasses qui apparaissent toujours au même endroit par rapport à la touche, quitte à avoir une distance légèrement plus longue à parcourir. Une autre conséquence du déplacement est le positionnement possible d'une KeyGlass juste au-dessus d'une autre touche ; ceci gêne alors le sujet dans le cas où il veut saisir la touche de dessous ou même s'il veut avoir la visibilité des deux (la touche de dessous et la KeyGlass) ;

- Enfin, les sujets ont trouvé déstabilisant qu'un caractère qui est apparu précédemment sur une KeyGlass de gauche par exemple, apparaisse peu de temps après sur une KeyGlass de droite. Ils aimeraient qu'un caractère qui soit apparu à une position donnée y reste lors des prédictions suivantes.

8.2 Seconde itération

8.2.1 Le système proposé

A partir de l'analyse des résultats et des discussions, nous avons réalisé une seconde version du prototype de notre système KeyGlass. Nous avons pris en compte dans cette seconde version, les deux premières remarques des sujets, évoquées à la section précédente.

Nous n'avons pas pris en compte la troisième remarque, car il semble difficile de pouvoir affecter une position particulière à chaque caractère sur les KeyGlasses. En effet, plus de quatre caractères sont régulièrement proposés par le système de prédiction. Leur attribuer une position fixe sur une KeyGlass poserait certains problèmes de conflits lorsque deux caractères auraient la même position privilégiée. Il serait cependant possible de prévoir des stratégies d'adaptation pour remédier à ces conflits, mais l'utilisateur ne serait que plus perturbé lorsque le caractère qu'il pense voir toujours apparaître à la même position, apparaîtrait à une position différente dans certains cas pour régler un conflit. Une conséquence possible pourrait être d'habituer l'utilisateur à une position particulière pour certains caractères ; l'utilisateur ne rechercherait alors plus le caractère aux autres positions qu'à celle couramment utilisée, et pourrait ne pas voir le caractère recherché dans le cas où il serait positionné ailleurs.

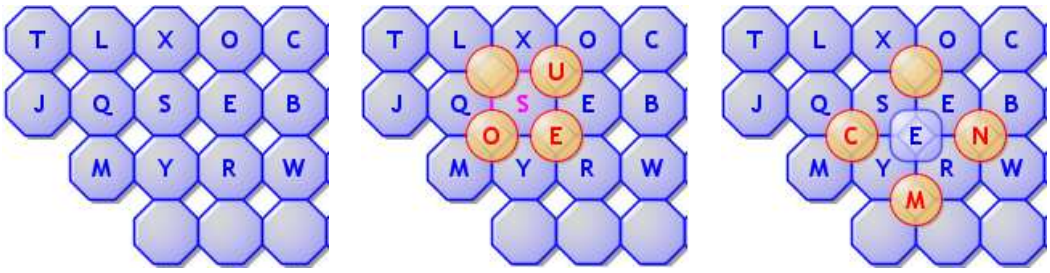


FIG. 43 – Design du clavier pour la version 2 des KeyGlasses : A. clavier simple ; B. KeyGlasses qui apparaissent après la saisie du 's' ; C. KeyGlasses qui apparaissent après la saisie du 'e' qui était placé sur une KeyGlass.

Pour remédier aux deux premiers problèmes, nous proposons un clavier de base dont les touches sont octogonales (cf. Figure 43 A.). Cette caractéristique donne l'avantage, lorsque les touches sont collées les unes aux autres, de laisser des espaces vides à la place des quatre coins traditionnels d'une touche rectangulaire. Nous avons utilisé ces espaces, comme seuls endroits autorisés pour l'affichage d'une KeyGlass. Ainsi, les KeyGlasses ne sont plus directement liées, au moment de la sélection d'une touche, aux coordonnées du pointeur de la souris, mais à la touche elle-même. Les KeyGlasses sont maintenant affichées dans les quatre espaces les plus proches de la touche qui vient d'être sélectionnée. Cette option règle ainsi le problème de recouvrement d'une touche par une KeyGlass. L'espace laissé libre par les touches octogonales évite le problème de lisibilité des caractères associés aux KeyGlasses (cf. Figure 43 B.). Enfin, pour régler le problème des doubles clics lors de la saisie d'une double lettre, lorsqu'une KeyGlass est sélectionnée, celle-ci reste active jusqu'à la sélection d'une autre touche, et quatre nouvelles KeyGlasses sont proposées sur les quatre espaces les plus proches (cf. Figure 43 C.).

Toujours dans la philosophie de diminuer au maximum les distances à parcourir, nous avons légèrement modifié la stratégie de proposition des KeyGlasses à afficher. Dans la première version du système, les quatre caractères les plus probables étaient systématiquement associés aux KeyGlasses. Avec le nouveau système de gestion du positionnement des KeyGlasses, celles-ci peuvent être légèrement plus éloignées que certaines touches. Ce cas peut, par exemple, se produire après avoir sélectionné une KeyGlass : par exemple, sur la Figure 43 C., après avoir saisi le 'e' de la KeyGlass (au moment de la configuration de la Figure 43 B.), les quatre nouvelles KeyGlasses sont légèrement plus éloignées que les touches fixes contenant les caractères 's','e','y' et 'r'. Nous avons choisi de filtrer, pour l'affichage sur les KeyGlasses, les caractères qui sont sur des touches plus proches que les KeyGlasses. Ainsi, pour l'exemple précédent, même si 's','e','y' ou 'r' faisaient partie des quatre caractères les plus probables, ils ont été supprimés de la sélection et c'est le caractère suivant dans le classement par probabilité qui a été choisi pour être affiché sur une KeyGlass.

8.3 Evaluation du système

Notre expérimentation est menée de manière à comparer principalement deux variables : d'une part, le système KeyGlass par rapport à un clavier logiciel seul ; d'autre part, les performances de sujets valides par rapport à des sujets handicapés des membres supérieurs.

8.3.1 Objectifs et hypothèses

La première hypothèse que nous avons formulée pour cette seconde expérimentation reste la diminution des distances lors de la saisie de texte. Est-ce que notre système tel que conçu dans la seconde version, avec des KeyGlasses proposées un peu plus éloignées que lors de la première évaluation, confirme une diminution des distances assez importante (de l'ordre de 50% en moins par rapport à un clavier classique) ?

Nous avons constaté lors de la première expérimentation, que la diminution des distances ne se traduisait pas directement par une diminution du temps de saisie. Comme nous l'avions également remarqué au travers des simulations par ordinateur, il semble y avoir un effet assez important de recherche visuelle des caractères qui engendre un temps supplémentaire pour la réalisation de la tâche. Ce temps de recherche visuelle diminue en théorie avec une utilisation régulière du système. Par conséquent, nous regarderons si le temps de saisie diminue au cours du temps, et s'il existe une différence sur l'effet d'apprentissage et le temps de recherche visuelle entre un clavier logiciel classique et le même clavier avec notre système KeyGlass.

8.3.2 Protocole expérimental

8.3.2.1 Le système à évaluer

Nous cherchons à comparer les caractéristiques d'un clavier logiciel seul ou couplé au système KeyGlass. Sachant que les sujets sont novices avec le système KeyGlass, nous souhaitons qu'ils aient le même profil pour le clavier logiciel seul. C'est pourquoi nous avons choisi de ne pas prendre le clavier logiciel AZERTY classique dont la disposition des caractères est connue de tous.

Nous avons créé un clavier logiciel identique au clavier logiciel AZERTY excepté par sa disposition des caractères. Nous avons gardé la barre espace en bas du clavier car différente des autres par sa forme. Néanmoins, celle-ci a été découpée en six touches (qui toutes permettent de saisir le caractère espace) pour coller au mieux au besoin de notre système. De plus, bien que différente, la disposition des caractères offre théoriquement à un utilisateur les mêmes performances de saisie que celle du clavier AZERTY : pour s'en assurer, nous avons utilisé le modèle de Soukoreff et Mackenzie pour évaluer la disposition des caractères. Ainsi, nous avons généré aléatoirement le positionnement des touches et, ce jusqu'à obtenir une valeur (donnée par la formule 9) équivalente à celle de la disposition AZERTY à 10^{-3} près. La figure 44 présente la disposition des caractères obtenue. C'est ce clavier qui est utilisé pour l'expérimentation. Pour éviter toute ambiguïté avec le clavier

logiciel AZERTY classique, nous appellerons ce clavier pour la suite du chapitre : le clavier SEB (du fait que ces trois lettres se suivent sur la seconde ligne).

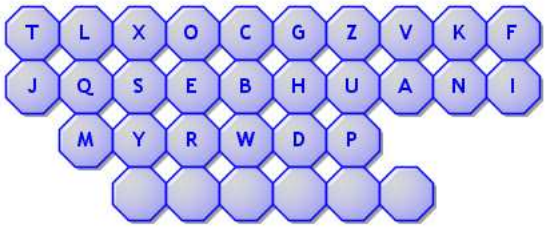


FIG. 44 – Clavier SEB : clavier de base utilisé pour l'expérimentation

8.3.2.2 Les sujets

Nous avons testé notre système sur deux populations de sujets : d'une part, des personnes ne présentant pas de handicap connu des membres supérieurs ; et d'autre part, des personnes ayant un handicap des membres supérieurs sans déficience cognitive *a priori*. L'objectif visé était de réunir 10 personnes de chaque profil. Pour cela, nous nous sommes adressés à l'Association Française contre la Myopathie (AFM) pour avoir un panel d'utilisateurs potentiels assez large. Cependant, sur la liste de 25 personnes qui nous a été fournie, seules 15 ont répondu favorablement à notre demande de participation à l'expérimentation, et seules six ont réussi à faire l'expérimentation dans son intégralité. Pour équilibrer, 10 personnes valides ont participé à l'expérimentation. Cependant, seules six ont fait l'expérimentation jusqu'au bout. Nous n'avons gardé les données que de 5 des 6 sujets valides du fait d'un problème sur les données du sixième.

8.3.2.3 Les variables

Les variables indépendantes de notre expérimentation sont :

- Les systèmes de saisie de texte que nous testons : le clavier SEB seul, et ce même clavier couplé au système KeyGlass ;
- Les sujets qui représentent deux profils différents : un handicap moteur des membres supérieurs ou non.

Les variables dépendantes que nous avons étudiées sont :

- Le taux de prédiction de caractères correct, obtenu lors de l'utilisation de notre système KeyGlass ;
- Le taux d'utilisation des caractères correctement prédits ;

- La distance parcourue par le pointeur du dispositif de pointage ;
- Le temps moyen mis pour saisir un mot ;
- Les erreurs de saisie.

8.3.2.4 La tâche de saisie

La tâche de saisie est la même que celle de la première expérience que ce soit pour la phase d'entraînement ou pour les exercices. Nous vous renvoyons à la section 8.1.3.1 pour plus de détails.

8.3.2.5 Le déroulement de l'expérimentation

Chaque sujet avait 20 sessions à réaliser. Chaque session consistait à effectuer une fois l'exercice de saisie avec le clavier SEB seul et une fois le même exercice avec ce même clavier couplé au système KeyGlass. Pour éviter certains effets d'apprentissage, l'ordre d'utilisation des claviers était contre balancé. Ainsi, une fois sur deux, le sujet commençait par le clavier seul et la fois suivante, il commençait par le clavier couplé au système KeyGlass. La moitié de chaque groupe de sujets a commencé les exercices en utilisant les claviers dans un certain ordre, et l'autre moitié dans l'autre ordre possible. De plus, l'exercice d'entraînement n'a eu lieu qu'avant le premier exercice de la première session.

Chaque sujet a réalisé l'ensemble de l'expérimentation à domicile. Les sujets étaient libres de réaliser une session de l'expérimentation quand ils le souhaitaient. Les seules contraintes qui leur avaient été données étaient de laisser au moins trois heures entre deux sessions d'expérimentation, et les sessions devaient être espacées d'au plus 48 heures.

8.3.2.6 Le matériel utilisé

Le système

L'ensemble du clavier et du système KeyGlass a été réalisé en JAVA. Le recueil des données a été réalisé avec la plate-forme E-ASSISTE. Pour chaque session d'expérimentation, l'ensemble des données recueillies était enregistré dans un fichier en respectant le langage KTML (cf. Chapitre 3, section 3.2.3.3 et l'Annexe A pour plus de détails).

L'interface de présentation des mots à recopier est la même que dans la première expérimentation (cf. section 8.1.3.1).

Le dispositif de pointage

Toutes les expérimentations ayant lieu chez les sujets, nous n'avons pu contrôler le materiel utilisé. Le dispositif de pointage utilisé a été le même pour chaque sujet valide. Par

contre, les sujets handicapés n'ayant pas tous la même déficience motrice, les dispositifs de pointage sont assez variés : deux sujets disposent d'une souris classique, deux d'une trackball, un d'un joystick et le dernier un touchpad.

8.3.3 Analyse des résultats

Voici la présentation des résultats obtenus à partir des données recueillies lors des 20 sessions des 11 sujets (6 sujets handicapés et 5 valides) ayant réalisé l'expérimentation jusqu'au bout et dans de bonnes conditions. Les résultats statistiques présentés dans cette section ont été obtenus par des tests paramétriques d'analyse de la variance (test ANOVA). Ces traitements ont été réalisés avec le logiciel SYSTAT 11.

Pour l'ensemble des résultats présentés ci-dessous : sur chaque histogramme, les valeurs du clavier SEB y sont représentées par les bâtons bleus. Pour les valeurs qui font intervenir le système KeyGlass, les bâtons sont couleur bordeaux. Pour les courbes de variation au cours des sessions, les courbes bleues, rouges et vertes représentent respectivement les valeurs pour l'ensemble des sujets, celles pour les personnes valides et celles des personnes handicapées.

8.3.3.1 Comparaison des claviers testés

Le tableau 15 présente les résultats généraux de cette expérimentation. Ces derniers représentent la moyenne sur l'ensemble des sessions de tous les sujets. Les différentes lignes du tableau correspondent à :

- Le *taux de prédiction* qui donne le pourcentage de caractères prédits correctement par notre système ;
- Le *taux d'affichage* qui donne le taux de caractères qui, étant dans les quatre plus fréquents à la sortie du système de prédiction, ont été affichés sur une KeyGlass ;
- Le *taux d'utilisation* qui donne sous forme de pourcentage, le taux d'utilisation des KeyGlasses par les utilisateurs. Ce taux est calculé à partir du nombre de fois où le caractère à saisir se trouve sur une des KeyGlasses par rapport au nombre de fois où l'utilisateur l'a utilisé ;
- La *Distance* qui est la distance moyenne (donnée en pixels) parcourue pour saisir un mot ;
- La *Durée* exprimée en millisecondes, traduit la durée moyenne dont a eu besoin l'utilisateur pour saisir un mot de l'expérimentation ;
- L'*Erreur* qui donne le KSPC, c'est-à-dire, le nombre de touches frappées divisé par le nombre de caractères correctement saisis.

Variables	SEB	SEB + KeyGlass
Taux de prédiction	-	69,11
Taux Affichage	-	90,22
Taux d'utilisation	-	96,02
Distance	1106	511
Durée	6895	8282
Erreur	1,015	1,024

TAB. 15 – Résultats généraux de l'expérimentation de la seconde itération

On peut ainsi constater que le taux de prédiction est un peu moins élevé que celui de la simulation. Il est important de mentionner que les utilisateurs ont fortement utilisé les KeyGlasses dès lors que cela était possible (plus de 96% d'utilisation). Ceci se traduit par une diminution des distances très significative ($p<0.001$) (plus de 53% de diminution entre le clavier SEB seul et ce même clavier avec le système KeyGlass). Par contre, la durée nécessaire pour saisir un mot reste encore plus élevée avec l'utilisation de notre système KeyGlass (il y a une augmentation d'environ 20% par rapport au clavier SEB seul). La différence de vitesse de saisie entre les deux claviers est aussi significative ($p<0.001$). Enfin, le taux d'erreur reste raisonnable pour les deux claviers avec moins de 2,5% d'erreurs pour le clavier SEB avec le système KeyGlass.

Voyons à présent si le type de population a une importance sur les différentes variables.

8.3.3.2 Discussion par rapport aux profils des sujets

Nous pouvons constater de manière générale sur les Figures 45 à 48, que les constats faits pour l'ensemble des sujets restent identiques pour les deux types de population qui ont participé à l'expérimentation.

Ainsi, nous pouvons voir que les KeyGlasses ont été utilisées à plus de 93% par les deux populations. Néanmoins, les sujets handicapés ont porté une attention plus particulière aux KeyGlasses avec un taux de 97,98% d'utilisation. L'écart type est même très faible, ce qui montre une très bonne implication de l'ensemble des sujets handicapés tout au long de l'expérimentation.

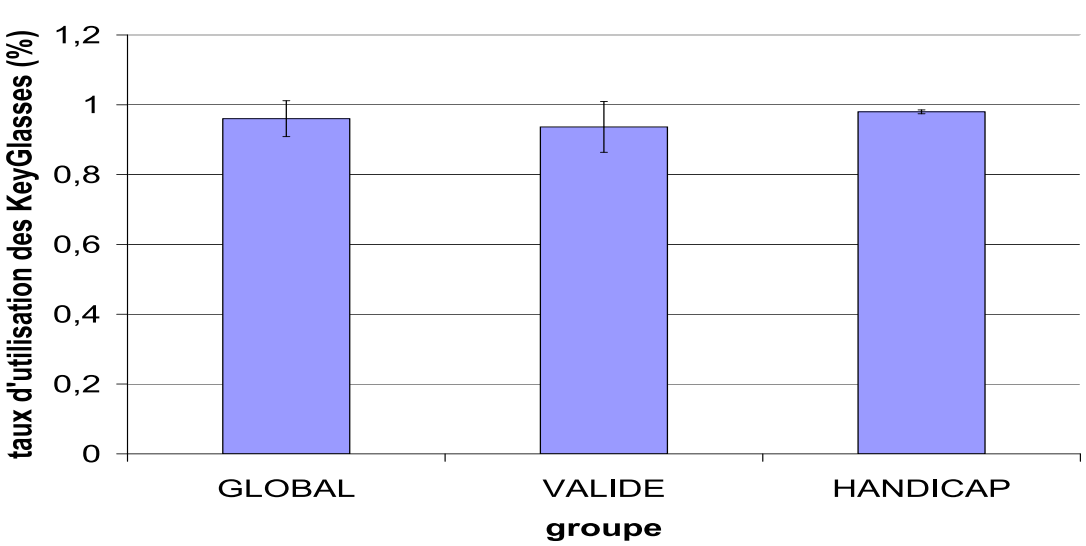


FIG. 45 – Taux d'utilisation des KeyGlasses

Cette forte utilisation se traduit par une minimisation des distances assez importantes. Les distances parcourues sont aussi homogènes pour les deux populations et sur les deux claviers.

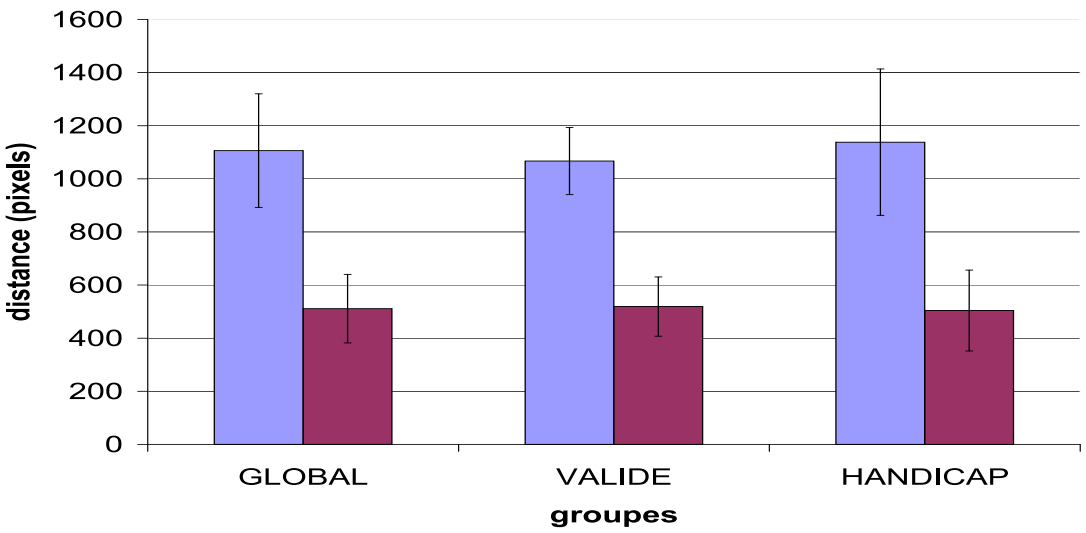


FIG. 46 – Distance

Si pour les temps, les deux populations obtiennent de meilleurs résultats avec le clavier SEB par rapport à son utilisation avec le système KeyGlass, on peut cependant constater un écart de performance assez important entre les deux populations. Les écarts types

montrent que le groupe de sujets valides est assez homogène, à l'inverse des sujets handicapés où l'écart type représente près de 50% de la valeur moyenne. Cet écart important peut être attribué à la fois aux différences de motricité des sujets et à l'hétérogénéité de l'ensemble de leurs dispositifs de pointage. Cette écart est aussi significatif d'après l'analyse de variance ($p<0.001$).

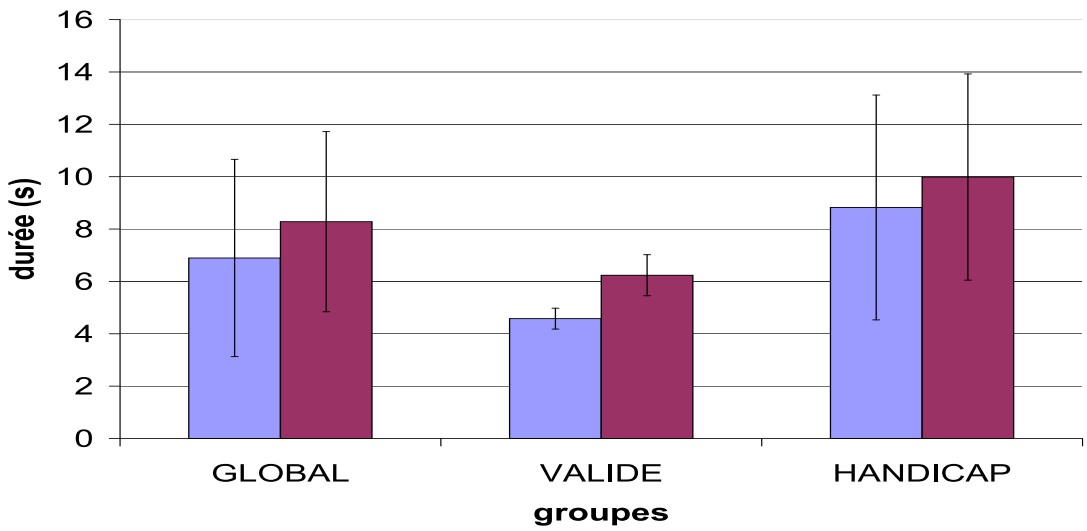


FIG. 47 – Durée

Comme nous pouvons le constater d'après les histogrammes représentant les durées de saisie d'un mot pour les différentes populations, pour le clavier SEB, les personnes handicapées ont besoin de presque deux fois plus de temps pour saisir un mot qu'une personne valide (4579ms pour les valides contre 8825ms pour les handicapés moteur soit une hausse de 92,73%). Pour le système KeyGlass associé au clavier SEB, les deux populations ont besoin de plus de temps pour saisir un mot, mais l'écart entre les deux populations diminue : 6236ms pour les valides contre 9987ms pour les handicapés, soit une augmentation d'environ 60,15%. Le système KeyGlass est par conséquent plus "bénéfique" aux personnes handicapées car même si le temps est encore plus long pour saisir un mot, l'augmentation par rapport au clavier logiciel seul est de 13,17% contre une augmentation de 36,18% pour les personnes valides.

Néanmoins, d'après la loi de Fitts, si l'on diminue la distance qui nous sépare de la cible à pointer, le temps pour atteindre cette dernière devrait diminuer. Or, comme nous venons de le voir, la distance diminue en moyenne de près de 54%, alors que la durée pour saisir un mot augmente en moyenne de 20%. Deux interprétations peuvent être avancées pour expliquer ce phénomène :

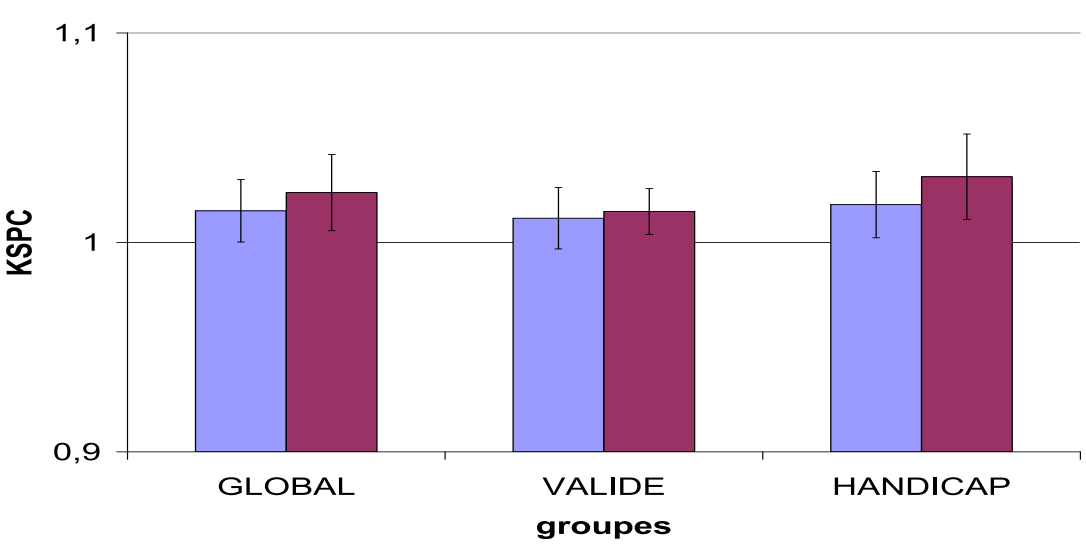


FIG. 48 – Erreur

- D'une part, les erreurs, qui peuvent être un frein pour la vitesse de saisie de texte lorsque celles-ci sont trop nombreuses. La Figure 48 montre que le nombre d'erreurs est un peu plus élevé lors de l'utilisation du système KeyGlass, mais reste cependant faible ;
- D'autre part, le temps de recherche d'un caractère, peut être un facteur important de perte de temps lors d'une tâche de saisie de texte, surtout lorsque les utilisateurs sont novices avec le système de saisie de texte. Pour estimer ce temps de recherche visuelle, nous avons calculé le temps de réaction de l'utilisateur. Nous définissons ce dernier, comme étant le temps qui sépare le moment où l'utilisateur a saisi un caractère et le moment suivant cette saisie où il amorce un mouvement avec le dispositif de pointage. Ainsi, nous considérons qu'à partir du moment où l'utilisateur commence à déplacer le pointeur, c'est qu'il a trouvé le caractère qu'il veut atteindre et qu'il commence le mouvement de pointage de celui-ci.

La Figure 49 présente les temps de réaction moyens. Il est intéressant de noter que ce temps de réaction est plus long lors des tâches de saisie sur le clavier SEB associé au système KeyGlass : 506 ms pour réagir lorsque les KeyGlasses sont présentées sur le clavier contre 302ms en moyenne sur le clavier logiciel seul (soit une augmentation de 67,5%). On peut interpréter ce temps supplémentaire comme étant celui nécessaire pour visualiser les quatre caractères qui se trouvent sur les KeyGlasses. On peut aussi voir sur cet histogramme que les personnes handicapées sont plus lentes pour amorcer un mouvement après avoir saisi un caractère.

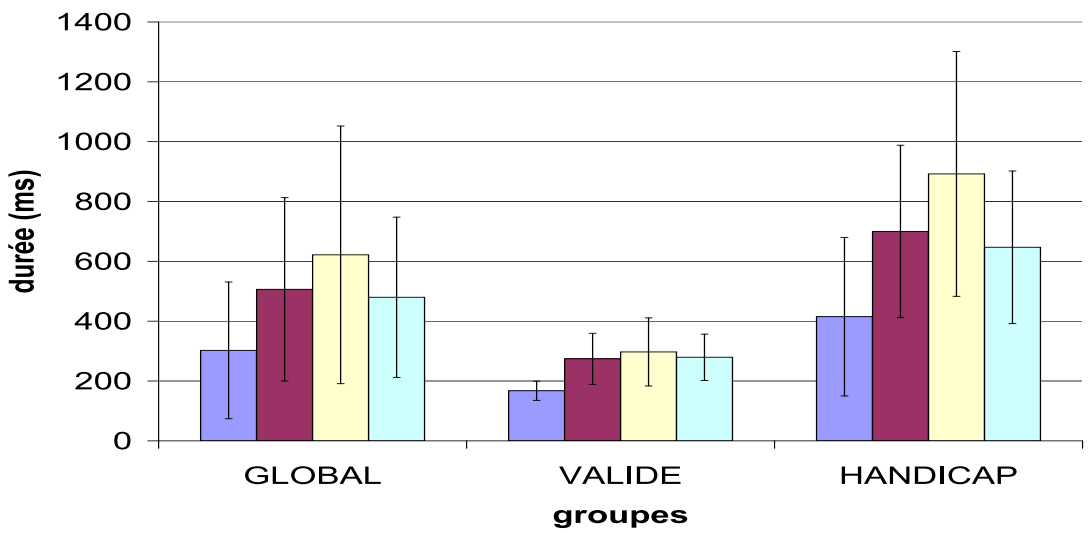


FIG. 49 – Temps Réaction

Enfin, nous avons aussi voulu savoir, pour le clavier SEB avec le système KeyGlass, s'il y avait une différence de temps de réaction entre un mouvement qui aboutirait à la saisie d'un caractère sur KeyGlass et celui qui aboutirait une touche fixe du clavier SEB. Ces deux temps de réaction sont représentés respectivement sur la figure 49 par les bâtons jaunes et verts pour la saisie d'un caractère sur une touche fixe et sur une KeyGlass. On peut ainsi constater que les sujets mettent plus de temps pour amorcer un mouvement quand il s'agit d'aller sur une touche fixe. Une hypothèse est qu'ils commencent par vérifier sur les quatre KeyGlasses que le caractère qu'ils cherchent ne s'y trouve pas, dans le cas contraire, ils vont rechercher sur les touches fixes.

Une des explications possibles de la perte de temps peut être le temps de recherche qui semble plus long lors de l'utilisation des KeyGlasses. Cependant, ce temps de recherche devrait normalement diminuer au cours des sessions d'appropriation (l'utilisateur maitrisant normalement de mieux en mieux le concept de KeyGlass). Les résultats que nous venons de présenter sont une moyenne sur l'ensemble des sessions. C'est pourquoi, nous allons maintenant regarder l'évolution des différentes variables au cours des sessions pour voir s'il existe un effet d'apprentissage.

8.3.3.3 Effet des sessions d'apprentissage

Les Figures 50 à 55 présentent l'évolution des différentes variables au cours des 20 sessions.

Concernant l'utilisation des KeyGlasses, nous pouvons voir que le taux d'utilisation a été dès la première session de l'ordre de 95%. Au départ, les deux populations utilisaient dans un même rapport les KeyGlasses. Par contre nous constatons qu'au cours des sessions, les sujets handicapés ont eu tendance à légèrement utiliser un peu plus les KeyGlasses alors que les sujets valides ont, au contraire, légèrement diminué leur utilisation de ces dernières. L'interprétation de ces résultats est que les sujets handicapés ont trouvé une plus grande utilité aux KeyGlasses (notamment du fait de la diminution des distances) que les sujets valides qui se sont peut être un peu découragés voyant que le système ne leur apportait pas énormément (du moins en vitesse de saisie de texte).

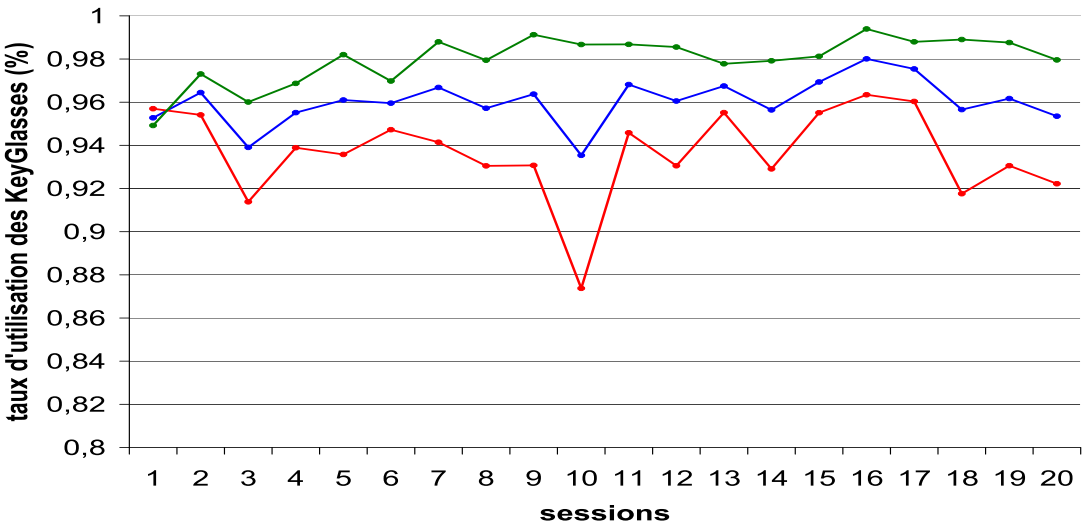


FIG. 50 – Prédiction par session

Les distances sont quant à elles très homogènes entre les deux populations (cf. Figure 51) qui ont exactement le même comportement au cours des sessions pour les deux claviers. Les courbes représentant les distances sur le clavier SEB sont en pointillés, alors que celles concernant ce même clavier avec le système KeyGlass sont en traits continus. Nous pouvons aussi voir que la distance sur le clavier SEB diminue de manière assez conséquente (-21%) lors des six premières sessions. Nous pouvons expliquer que cette diminution de distance sur ce clavier, qui pourtant reste fixe, est en partie due à la culture des sujets qui ont l'habitude d'utiliser un clavier AZERTY et qui anticipaient leur mouvement vers des caractères en pensant les trouver à la même place qu'habituellement sur le clavier AZERTY.

Pour justifier cette hypothèse, la figure 52 illustre deux exemples de tracés rejoués qui montrent des départs de mouvements contradictoires avec ceux qui doivent être réellement

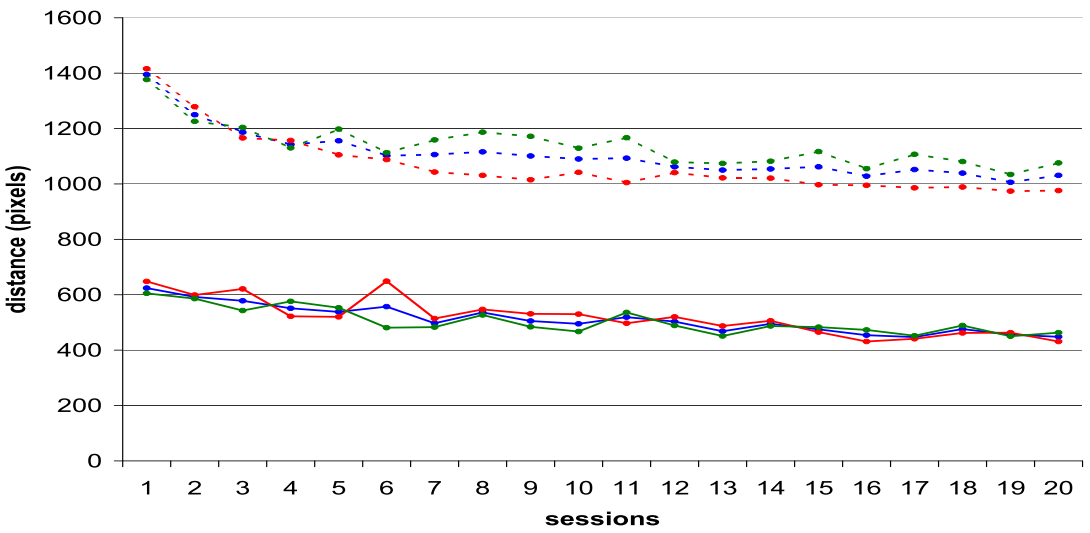
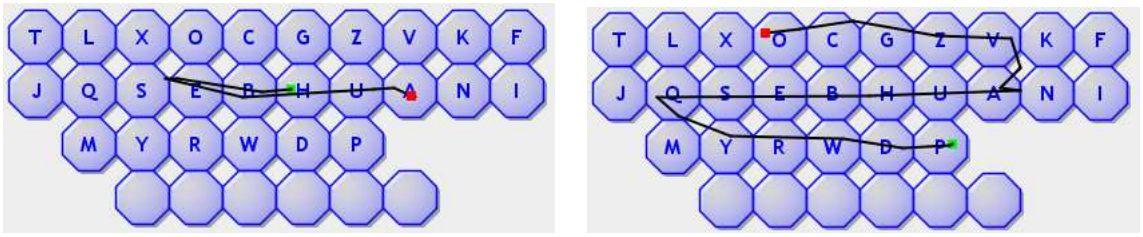


FIG. 51 – Distance par session

effectués. La Figure 52 A. présente le déplacement du caractère 'H' au caractère 'A'. Nous remarquons que le tracé part initialement en sens inverse du 'A'. Ce tracé a été recueilli lors de la première session d'expérimentation d'un sujet. Nous pensons que celui-ci a anticipé le déplacement vers le 'A' qui se trouve habituellement en haut à gauche du clavier AZERTY. Quant à la figure 52 B., elle présente le déplacement du caractère 'P' vers le caractère 'O'. On constate que l'utilisateur a effectué de grands mouvements de gauche à droite avant de pointer le 'O'. Une explication possible est ici que l'utilisateur a déplacé son pointeur pour rechercher le caractère 'O'.



A. B.

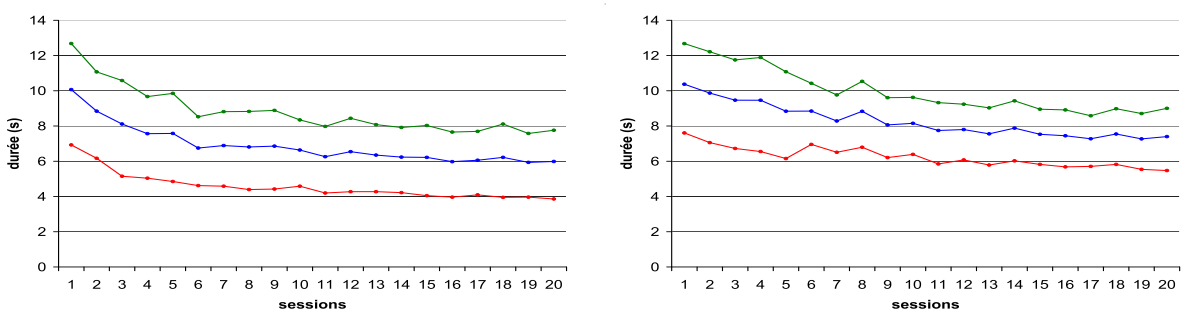
FIG. 52 – Exemple de tracés ayant entrainé une augmentation de la distance

De même que pour la distance, nous pouvons voir pour les courbes de temps (cf. Figure 53 A. et B.) que l'évolution au cours des sessions est indépendante du profil du sujet.

La figure 53 A. présente l'évolution de la durée nécessaire pour saisir un mot sur le clavier SEB. La diminution importante de la distance que nous avons constatée sur les

six premières sessions se traduit aussi par une diminution du temps. Cette diminution est même plus importante au niveau du temps : elle représente un gain sur la durée de 33% entre les sessions 1 et 6. Dans le même intervalle, le gain de temps sur ce clavier avec le système KeyGlass n'est que de 20,11%. Une fois ces six ou sept premières sessions passées, le gain jusqu'à la vingtième session est moins important (12%) quel que soit le clavier utilisé (le clavier SEB avec ou sans le système KeyGlass).

Nous pensons que l'essentiel de l'apprentissage de la position des caractères sur un clavier se fait pendant les dix premières sessions. Il apparaît aussi qu'il est plus facile d'apprendre une disposition de caractères fixes qu'une disposition de caractères dynamiques qui varie en fonction de la saisie. Cependant, cette seconde étant plus compliquée à "apprendre", on pourrait penser que l'apprentissage continuerait pendant plus longtemps que pour une disposition de caractères fixes. Or, au regard des courbes A et B de la Figure 53, il apparaît que pour les deux claviers, la courbe d'apprentissage est la même.



A) B)

FIG. 53 – Vitesse par session : A. pour le clavier SEB ; B. avec le système KeyGlass

Pour avoir une vue plus globale du comportement des sujets, les figures 54 et 55 présentent les temps de réaction au cours du temps. Une fois de plus, le comportement des deux populations est identique. Le temps de réaction des personnes handicapées est plus élevé, mais les courbes d'évolution sont identiques à celles des personnes valides.

Sur la figure 54, nous présentons les temps de réaction pour le clavier SEB seul (figure A.) et pour le clavier SEB associé au système KeyGlass (cf. Figure B.). Sur les courbes de la figure A., nous pouvons voir que le temps de réaction n'a pas de diminution aussi significative lors des 6 ou 7 premières sessions que celle que nous avons constaté pour la distance et le temps. Ceci renforce l'hypothèse formulée au moment de l'interprétation des distances, en évoquant une possible anticipation du mouvement en pensant connaître le clavier. Les sujets se représentant une disposition du clavier AZERTY anticiperaient un mouvement, et le modifieraient en cours de déplacement. Ceci expliquerait que les temps

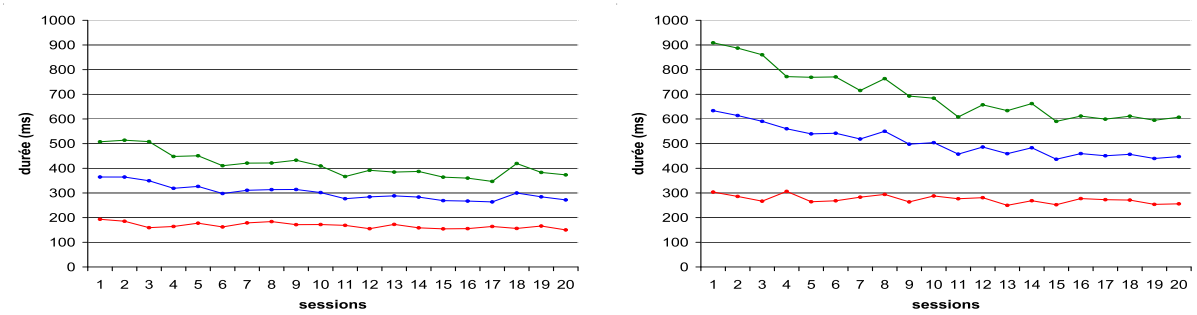


FIG. 54 – Temps de réaction de chaque clavier

de réaction soient assez faibles dès la première session et que la distance soit plus élevée lors des 6 ou 7 premières sessions.

Par contre, nous constatons que le temps de réaction est bien plus élevé au départ avec les KeyGlasses, et que la diminution du temps de réaction sur les 6 ou 7 premières sessions est plus prononcée (avec près de 20% de temps en moins après 7 sessions). Il apparaît par conséquent que l'affichage des KeyGlasses "oblige" l'utilisateur à regarder les caractères présents sur celles-ci et supprime l'effet d'anticipation après la saisie d'un caractère.

Nous remarquons que bien qu'un peu plus lent pour amorcer son geste, un sujet valide reste constant dans ces temps de réaction : il n'y a pas d'effet d'apprentissage.

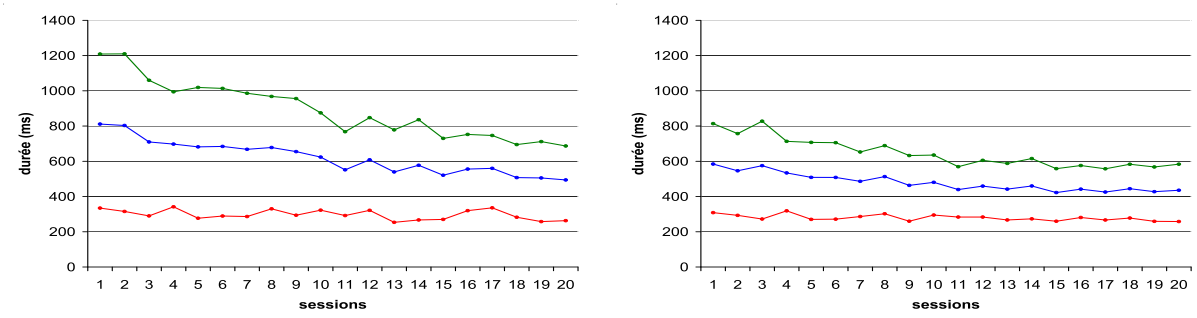


FIG. 55 – Temps de réaction spécifique sur le clavier SEB associé au système KeyGlass

Enfin, la figure 55 présente la décomposition des temps de réaction lors de l'utilisation du système KeyGlass avec le clavier SEB. Les courbes des figures A. et B. présentent l'évolution du temps de réaction moyen qui aboutissent à la saisie d'un caractère respectivement sur une touche fixe du clavier SEB et sur une KeyGlass. On peut ainsi constater que lors des premières sessions, c'est le temps de réaction pour aller saisir un caractère

sur une touche du clavier SEB qui est le plus pénalisant (environ 800ms contre 600ms pour le temps de réaction avant la saisie sur une KeyGlass). La diminution de cette durée est par ailleurs plus importante pour la saisie avec le clavier SEB que pour celle avec le système KeyGlass (23,18% contre 17,8%). On peut supposer que cette diminution est due en partie à l'effet d'apprentissage : d'une part, le sujet sait au bout de quelques sessions s'il va trouver (ou non) sur les KeyGlasses, le caractère qu'il veut saisir. S'il sait que le caractère n'y est pas, il va changer sa stratégie en cherchant directement sur le clavier fixe le caractère voulu. Nous voyons que l'effet d'apprentissage s'applique aussi au clavier SEB ; ce temps de recherche diminue.

8.4 Discussion

A partir de ces résultats et des constatations que nous avons pus en déduire, il apparaît que le système KeyGlass est bien perçu par les personnes handicapées. Notre première hypothèse qui était de diminuer les distances est largement atteinte.

Par contre, malgré cette diminution des distances, le temps mis par les sujets pour saisir un mot demeure plus long lorsqu'ils utilisent le système KeyGlass. L'hypothèse que nous faisions concernant la diminution du temps au cours des sessions n'est que partiellement vérifiée. En effet, le temps de saisie diminue lors des premières sessions, mais l'apprentissage n'a pas plus d'effet que pour un clavier logiciel classique.

Après discussion informelle avec les sujets, il apparaît que si le principe de rajouter des touches à proximité est grandement apprécié, le fait que les caractères changent en fonction de la saisie du mot les perturbe énormément dans leur vitesse de saisie de texte. C'est certainement la trop grande variabilité des caractères proposés qui entraînerait cette perte de temps sur le système KeyGlass.

Perspectives

Au regard des résultats des deux premières itérations et après discussion avec les utilisateurs finaux (les personnes handicapées), le système KeyGlass apporte bien une forte diminution des distances qui est appréciable pour ces personnes. Néanmoins, le fait de proposer sur ces touches les caractères les plus probables ne semble pas être la meilleure stratégie, du moins pour gagner en vitesse de saisie de texte.

C'est pourquoi, nous envisageons maintenant de faire de nouveaux tests en modifiant cette fois-ci la proposition des caractères associés aux KeyGlasses. L'objectif est de permettre à l'utilisateur d'apprendre les caractères qui se trouvent sur les KeyGlasses pour que les utilisateurs puissent anticiper leur mouvement. Nous envisageons de tester deux nouvelles stratégies de proposition de caractères :

- Proposer quel que soit le caractère saisi, toujours les mêmes caractères sur les KeyGlasses (comme c'est le cas par exemple sur le *marking-menu* proposé par Isokoski [Isokoski 04b]). Cette solution diminuera le nombre de fois où il est possible d'utiliser les KeyGlasses, mais c'est la méthode la plus adaptée pour mémoriser ces caractères supplémentaires ;
- L'autre solution est un intermédiaire entre ce que nous avons déjà réalisé et la solution d'Isokoski. Elle consiste à proposer pour un caractère donné toujours le même ensemble de caractères prédits. Ainsi pour chaque caractère, il y aura une configuration de caractères différente. Cela fait un peu plus de configurations à apprendre, mais il paraît possible de les apprendre.

Ceci nous permettra de tester ces deux possibilités pour voir quel est le meilleur compromis entre la réduction de distance, le temps gagné et le temps d'apprentissage.

Une fois que nous aurons décidé de la stratégie à adopter pour les caractères associés sur les KeyGlasses, il serait intéressant de tester notre système KeyGlass sur un clavier logiciel complet. La limitation que nous avons faite aux 26 caractères de l'alphabet latin et à l'espace, n'est pas représentatif de la saisie de texte pour la langue française. C'est pourquoi la dernière étape de notre processus itératif sera la conception et l'évaluation de notre système sur un clavier complet.

Conclusion et perspectives

Conclusion

Les recherches sur la saisie de texte au travers des moyens logiciels (claviers logiciels) concernaient en grande partie l'accessibilité de l'informatique pour les personnes handicapées. Avec l'émergence des dispositifs mobiles, les recherches dans ce domaine se sont considérablement accrues depuis une décennie avec l'objectif de pallier le manque de clavier physique sur les dispositifs de petite taille tout en gardant un confort et des performances de saisie semblables à celles qu'il est possible d'obtenir sur un clavier physique d'ordinateur de bureau.

Bien que ces deux contextes d'applications aient le même problème (la non accessibilité à un clavier physique) et le même besoin (pouvoir saisir du texte sur leur dispositif), les moyens d'interaction mis à leur disposition ne sont pas identiques : stylet et écran tactile d'une part, contre dispositif de pointage et généralement clavier logiciel. Cette différence, ajoutée aux caractéristiques différentes des utilisateurs des deux situations, fait qu'il est difficile de concevoir un système qui soit utile et utilisable par tous.

Or, les nombreux travaux qui ont été réalisés ces dernières années pour l'optimisation et l'amélioration de la saisie de textes, ont été conçus principalement pour une utilisation sur dispositif mobile, sans toujours se soucier de son utilisation par une personne handicapée. C'est pourquoi nous avons choisi de focaliser ces travaux de thèse sur la conception de claviers logiciels orientés sur la problématique du handicap, et plus particulièrement sur la minimisation des mouvements à effectuer pour saisir du texte.

Nous avons pour cela étudié deux axes de recherche en parallèle :

- Le premier consiste à réorganiser les touches sur le clavier et les caractères de manière à diminuer l'amplitude des déplacements effectués entre les caractères à saisir consécutivement. Devant la complexité du problème à résoudre, nous avons proposé d'appliquer les algorithmes génétiques à notre problème. Les résultats obtenus à partir de cette méthode apportent de meilleurs résultats que ceux d'autres techniques d'optimisation. La réduction de distance que nous obtenons peut même se traduire par un gain en temps selon les lois prédictives.

- Le second consiste à apporter des touches supplémentaires à proximité de la dernière frappée et d'y associer les caractères qui ont le plus de probabilité d'être saisis. Ces travaux s'inspirent d'autres réalisés pour une utilisation sur dispositifs mobiles alliant saisie sur clavier logiciel et gestes de désignation. Nous avons choisi de supprimer cette partie gestuelle, et nous nous sommes plus focalisés sur la minimisation des distances. Pour cela, les caractères que nous proposons en plus sont prédits par un système de prédiction alliant table bigramme et arbre lexicographique. Cette prédiction apporte sur une KeyGlass le caractère à saisir dans plus de 70% des cas. Ceci se traduit par une diminution de plus de 50% de la distance à parcourir.

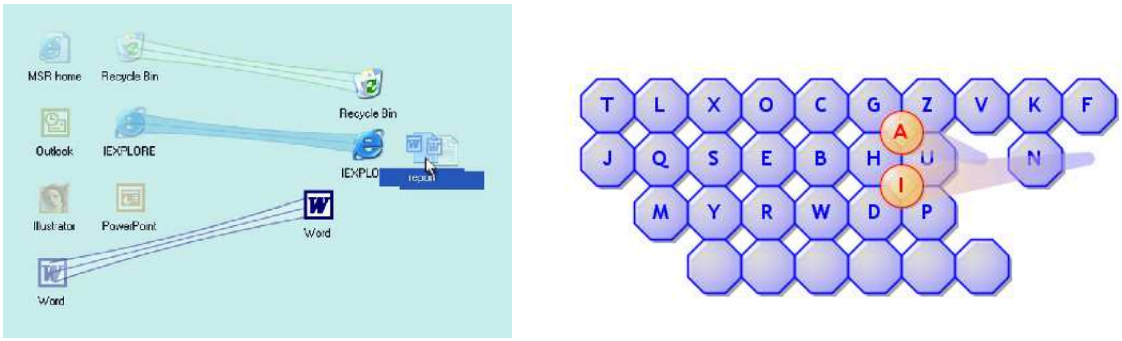
Si nous avons atteint notre objectif de diminution des distances, notre système dans sa version actuelle, ne permet pas d'augmenter la vitesse de saisie de texte. Une des raisons que nous pouvons avancer est le fait que les caractères proposés sur les KeyGlasses varient trop souvent. Ceci empêche l'utilisateur d'apprendre la disposition des caractères sur ces KeyGlasses et donc laisse ce dernier dans une situation de novice tout au long de la saisie. Il est alors nécessaire de trouver un bon compromis entre la réduction de distance et la prédiction des caractères. En effet, des caractères qui bougeraient moins sur les KeyGlasses permettraient à l'utilisateur de les retenir, par contre la distance parcourue lors de la saisie serait un peu plus longue.

Enfin, notons que l'ensemble des expérimentations liées au système KeyGlass ont été réalisées à partir de la plate-forme E-Assiste. Ceci nous a permis de développer un certain nombre d'outils pour l'expérimentation, qui sont maintenant réutilisables pour d'autres expérimentations. Nous avons commencé à les réutiliser pour l'expérimentation de nouveaux claviers dans le cadre du projet ChatCom.

Perspectives

En 2004, nous fêtions le 50^e anniversaire de la loi de Fitts [Guiard 04]. A cette occasion, Balakrishnan [Balakrishnan 04] réalisa une revue des divers travaux entrepris pour faciliter la tâche de pointage d'une cible. Rappelons que cette tâche de pointage est prépondérante dans la saisie de textes via un clavier logiciel. Or, parmi l'ensemble des propositions dont il fait état, très peu ont fait l'objet d'une application à la saisie de textes. C'est pourquoi, nous proposons à travers ces perspectives des idées d'applications aux claviers logiciels, que nous avons pour la plupart déjà prototypés, mais dont il nous reste à en montrer l'utilisabilité.

Baudisch et ses collègues [Baudisch 03] proposèrent en 2003 l'idée du Drag-and-Pop : celui ci consiste au moment d'effectuer un Drag-and-Drop à rapprocher les cibles qui sont susceptibles d'être celles que souhaite atteindre l'utilisateur. Par exemple, lorsque celui-ci commence un Drag-and-Drop avec un fichier Word, on peut penser que c'est soit pour le supprimer en le mettant à la poubelle, soit pour l'afficher avec Word ou Internet Explorer (cf. Figure 56 A.). Dans ce cas là, le Drag-and-Pop avance vers l'objet déplacé par l'utilisateur les icônes des applications qui peuvent l'intéresser.



A) B)

FIG. 56 – Le principe Drag-and-Pop : A. Principe proposé par Baudisch et al. pour un bureau de PC [Baudisch 03] ; B. Principe appliqué à un clavier logiciel. Par exemple, après la saisie d'un 'b', on peut imaginer rapprocher les caractères 'a' et 'i'.

Ainsi, il serait intéressant d'imaginer un clavier à l'intersection entre les KeyGlasses et le Drag-and-Pop (cf. Figure 56 B.). Au lieu de rajouter de nouvelles touches au fur et à mesure de la saisie, on pourrait par exemple rapprocher les plus probables. Celles-ci seraient choisies après que l'utilisateur ait effectué le début de son geste. Ceci permettrait de connaître la direction dans laquelle il souhaite aller, et ainsi limiter le déplacement des touches (contenant les caractères les plus probables) qui se trouvent dans cette direction. La Figure 56 B., montre un exemple de ce qui pourrait se passer après la saisie du caractère 'b', où les caractères 'a' et 'i' sont rapprochés car leur fréquence d'apparition après un 'b' est assez élevée et ils se trouvent à l'origine assez loin du 'b' sur le clavier.

McGuffin et Balakrishnan [McGuffin 02] ont eux proposé l'extension des cibles : c'est-à-dire augmenter la taille d'une cible quand on se rapproche de celle-ci avec le pointeur ou au contraire diminuer sa taille dès que l'on s'en éloigne (cf. Figure 57). Cependant, il a été démontré depuis par Zhai et ses collègues [Zhai 03a] que cette extension des cibles ne permettait pas de diminuer la distance à parcourir pour aller pointer une cible, et que cela était identique à laisser les cibles inchangées.

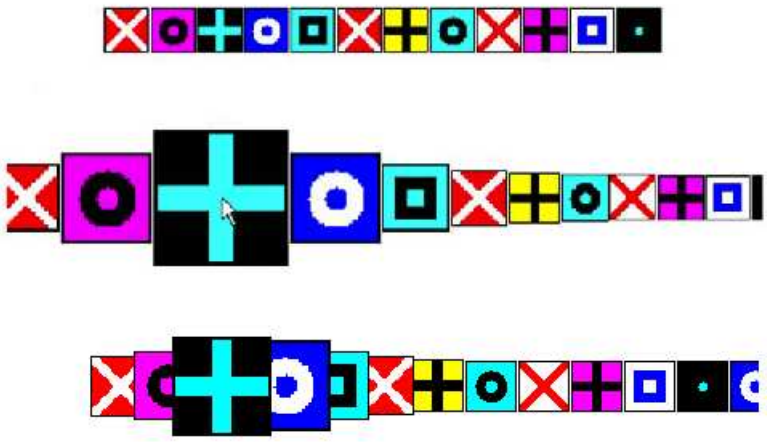
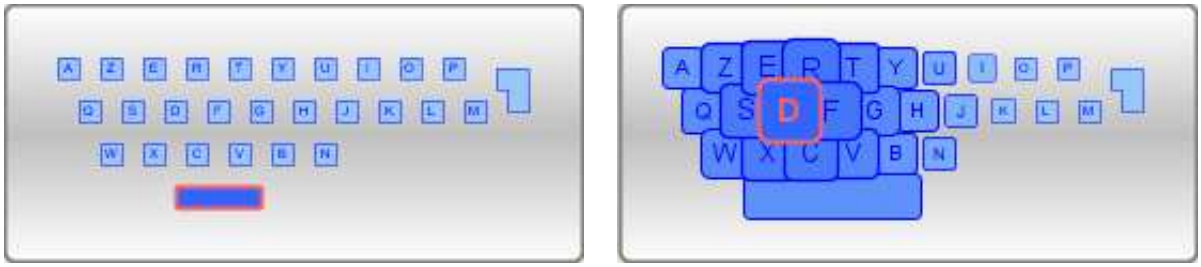


FIG. 57 – Extension de cibles : Plus le pointeur approche, plus la cible grossie.

Néanmoins, on peut imaginer appliquer ce principe à un clavier logiciel. Si le fait d'appliquer telle quelle l'extension de cible n'apporte rien en réduction de distance, on pourrait utiliser ce principe pour diminuer la taille des claviers (cf. Figure 58). Ce clavier serait spécifique à une utilisation par des experts. Les touches seraient réduites au point de ne pas voir les caractères qui y sont associés (cf. Figure 58 A.), mais un utilisateur expert connaissant parfaitement l'agencement de son clavier, n'aurait pas besoin de voir les caractères à l'avance. Il se déplacerait directement dans la direction du caractère

recherché. Les touches seraient alors agrandies lors du passage du pointeur à proximité ce qui permettrait de voir les caractères et de sélectionner le bon à ce moment là.



A) B)

FIG. 58 – Le clavier à touches extensibles : A. clavier réduit sans que le pointeur ne se trouve dessus ; B. Clavier déformé par la présence du pointeur à proximité du 'D'

Une autre proposition serait de supprimer la flèche qui sert généralement de curseur pour le dispositif de pointage et de la remplacer par la sélection automatique de la touche la plus proche sur le clavier. Ceci ne s'appliquant que quand le pointeur entre sur la zone du clavier logiciel. Cette proposition présente deux avantages : à la manière du Bubble-Cursor [Grossman 05], une touche serait toujours sélectionnée une fois le pointeur sur le clavier, ceci évitant de positionner le curseur dans le vide entre deux touches. Ce clavier sans pointeur est à l'heure actuelle testé dans le cadre du projet ChatCom.

On peut aussi imaginer un clavier appliquant le pointage sémantique défini par Blanch et ses collègues [Blanch 04]. Le pointage sémantique consiste à passer plus rapidement sur les zones de l'écran où il n'y a pas ou peu d'informations pertinentes. Ce principe pourrait être appliqué au clavier logiciel où le pointeur passerait rapidement sur les touches où le caractère associé a peu de chance d'être saisi, et passerait plus longtemps sur les touches qui ont le plus de chance d'être frappées.

Annexes

A

Protocole de communication de la plate-forme E-Assiste

A.5 Types de messages autorisés

Voici l'ensemble des messages qui sont permis sur la plate-forme E-Assiste. Nous présentons pour chaque type de message, d'une part la version balisée qui est utilisée pour la sauvegarde de données au format KTML, et d'autre part le type de message Ivy qui est envoyé sur le bus logiciel pour la communication temps réel.

A.5.1 Début d'une évaluation

Message qui annonce le début d'une expérimentation. Ce message donne le nom du sujet qui réalise cette expérimentation et quel système il teste. Cette balise KTML est la seule à être ouvrante / fermante. La balise ouvrante commence le fichier KTML et la balise fermante clôture ce fichier. Tout ce qui est en dehors de ces balises n'est pas pris en compte dans l'analyse. La balise ouvrante Evaluation initialise toujours le temps à 0. Toutes les autres balises ont un attribut temps 't'. Ce temps représente le temps qui s'est écoulé en millisecondes depuis le lancement de l'expérimentation.

Balise KTML :

<Evaluation syst="nomSyst" sujet="nomSujet" t="0">

Message Ivy :

E-Assiste :Evaluation syst=nomSyst sujet=nomSujet t=0

A.5.2 Début d'un exercice

Nous avons défini ce message pour annoncer le passage d'un exercice à un autre. Le type de l'exercice est défini par les concepteurs, il n'y a pas de types prédéfinis.

Balise KTML :

<DebutExercice type="typeExercice" t="t"/>

Message Ivy :

E-Assiste :DebutExercice type=typeExercice t=t

A.5.3 Données à recopier

Deux messages ont été définis pour connaître à tout moment ce que doit recopier le sujet dans une tâche de recopie : un message pour les mots et un pour le texte.

Balise KTML :

<MotACopier mot="mot" t="t"/>

<TexteACopier texte="texte" t="t"/>

Message Ivy :

E-Assiste :MotACopier mot=mot t=t

E-Assiste :TexteACopier texte=texte t=t

A.5.4 Caractère saisi

Ce message définit le caractère qui est saisi et de quel type de support il provient. Deux types de support sont déjà définis : *fixe* qui définit que le caractère a été saisi à partir d'une touche appartenant au clavier logiciel de base, et *keyglass* lorsque le caractère a été saisi à partir d'une touche ajoutée par le système KeyGlass .

Balise KTML :

<TextInput string="caractere" type="type" t="t"/>

Message Ivy :

E-Assiste :TextInput string=caractere type=type t=t

A.5.5 Evenements du dispositif de pointage

Les différents événements produits par la souris -lorsque celle-ci se trouve sur le clavier logiciel testé- sont capturés et renvoyés sous forme de message. Tous les événements n'ont pour le moment pas été capturés et renvoyés. Nous n'avons traité que ceux qui nous intéressaient dans nos analyses : c'est-à-dire ceux de déplacement du pointeur, et de pression / relâchement d'un des boutons du dispositif de pointage.

A.5.5.1 Déplacement du pointeur

En plus des coordonnées (x,y) du pointeur à l'écran, nous donnons une information pour savoir si ce pointeur n'a pas subi une transformation d'échelle entre le déplacement qui est réellement effectué avec le dispositif de pointage et le déplacement du pointeur à l'écran. Cette information est relayée par la variable *identique* qui définit si oui ou non les deux déplacements sont identiques.

Balise KTML :

<MouseMoved x="x" y="y" identique="flags" t="t"/>

Message Ivy :

E-Assiste :MouseMoved x=x y=y identique=flags t=t

A.5.5.2 Pression d'un bouton

Ce message donne les coordonnées (x,y) du pointeur au moment du clic. Lorsque le dispositif de pointage comporte plusieurs boutons, il est possible de les différencier grâce à la variable *type*. Lors de l'expérimentation, il est aussi attribué un identifiant *id* à chaque clic.

Balise KTML :

<MousePressed x="x" y="y" t="t" type="type" id="idPos"/>

Message Ivy :

E-Assiste :MousePressed x=x y=y t=t type=type id=idPos

A.5.5.3 Relâchement d'un bouton

Balise KTML :

<MouseReleased x="x" y="y" t="t"/>

Message Ivy :

E-Assiste :MouseReleased x=x y=y t=t

A.5.6 Touche pressée

Ce message désigne la touche qui a été pressée. Les coordonnées (x,y) correspondent à la position de la touche sur le clavier (souvent définie soit par le coin supérieur gauche de la touche, soit par son centre). Ce message donne aussi le caractère qui est associé à cette touche. Un identifiant est associé à chaque touche frappée lors de l'expérimentation.

Balise KTML :

<KeyPressed x="x" y="y" t="t" id="idPos" char="caractere"/>

Message Ivy :

E-Assiste :KeyPressed x=x y=y t=t id=idPos char=caractere

A.5.7 Prédiction de caractères

Voici le type de message renvoyé par le système de prédiction de caractères. La variable *isend* définie par un booléen si ce qui a été saisi auparavant peut être un mot à part entière. La variable *classement* renvoie un classement des caractères triés par ordre décroissant par rapport à la probabilité d'apparaître après ce qui vient d'être saisi. Chaque caractère est séparé par ' :'.

Balise KTML :

<Prediction isend="isEnd" classement="classementLettre" t="t" id="idPred"/>

Message Ivy :

E-Assiste :Prediction isend=isEnd classement=classementLettre t=t id=idPred

A.5.8 Gestion des KeyGlasses

Deux messages ont été créés plus particulièrement pour la gestion des KeyGlasses.

A.5.8.1 Affichage des Keyglasses

Ce message annonce l'affichage d'une touche supplémentaire avec le caractère à y associer, ainsi que la position à laquelle on doit la positionner. Les coordonnées (x,y) sont celles du coin supérieur gauche de la touche.

Balise KTML :

<KeyGlass x="x" y="y" t="t" char="caractere"/>

Message Ivy :

E-Assiste :KeyGlass x=x y=y t=t char=caractere

A.5.8.2 Initialisation des KeyGlasses

Ce message annonce l'initialisation des KeyGlasses. Celle-ci a lieu à la fin de chaque mot, dès que le sujet saisit un caractère espace.

Balise KTML :

<KeyGlassInit t="t" id="idPred"/>

Message Ivy :

E-Assiste :KeyGlassInit t=t id=idPred

A.5.9 Modification de variable

Dans certains cas, une variable pouvant être modifiée (pendant la phase de prise en main par exemple), un message a été créé à cet effet. Le nom de la variable et sa nouvelle valeur sont des attributs.

Balise KTML :

<VariableChange nom="nomVar" valeur="valeurVar" t="t"/>

Message Ivy :

E-Assiste :VariableChange nom=nomVar valeur=valeurVar t=t

A.5.10 Initialisation de l'horloge

Pour que tous les agents puissent être synchronisés, l'heure de départ leur est donnée en millisecondes par l'agent principal qui lance l'expérimentation.

Balise KTML :

<InitHorloge t="t"/>

Message Ivy :

E-Assiste :InitHorloge t=t

A.6 DTD du langage KTML

```
<!ELEMENT Evaluation (DebutExercice|InitHorloge|MousePressed
|MouseReleased|MouseMoved|Prediction|TextInput|MotACopier
|KeyPressed|KeyGlass|KeyGlassInit)*>

<!ELEMENT DebutExercice EMPTY>
<!ATTLIST DebutExercice type CDATA #REQUIRED>
<!ATTLIST DebutExercice t CDATA #REQUIRED>

<!ELEMENT InitHorloge EMPTY>
<!ATTLIST InitHorloge t CDATA #REQUIRED>

<!ELEMENT MousePressed EMPTY>
<!ATTLIST MousePressed x CDATA #REQUIRED>
<!ATTLIST MousePressed y CDATA #REQUIRED>
<!ATTLIST MousePressed t CDATA #REQUIRED>
<!ATTLIST MousePressed type CDATA #REQUIRED>
```

```
<!ATTLIST MousePressed id CDATA #REQUIRED>

<!ELEMENT MouseReleased EMPTY>
<!ATTLIST MouseReleased x CDATA #REQUIRED>
<!ATTLIST MouseReleased y CDATA #REQUIRED>
<!ATTLIST MouseReleased t CDATA #REQUIRED>

<!ELEMENT MouseMoved EMPTY>
<!ATTLIST MouseMoved x CDATA #REQUIRED>
<!ATTLIST MouseMoved y CDATA #REQUIRED>
<!ATTLIST MouseMoved t CDATA #REQUIRED>
<!ATTLIST MouseMoved identique CDATA #REQUIRED>

<!ELEMENT prediction EMPTY>
<!ATTLIST prediction p1 CDATA #IMPLIED>
<!ATTLIST prediction p2 CDATA #IMPLIED>
<!ATTLIST prediction p3 CDATA #IMPLIED>
<!ATTLIST prediction p4 CDATA #IMPLIED>
<!ATTLIST prediction isend CDATA #IMPLIED>
<!ATTLIST prediction classement CDATA #IMPLIED>
<!ATTLIST prediction id CDATA #IMPLIED>
<!ATTLIST prediction t CDATA #REQUIRED>

<!ELEMENT textInput EMPTY>
<!ATTLIST textInput t CDATA #REQUIRED>
<!ATTLIST textInput string CDATA #REQUIRED>
<!ATTLIST textInput type CDATA #REQUIRED>

<!ELEMENT KeyPressed EMPTY>
<!ATTLIST KeyPressed x CDATA #REQUIRED>
<!ATTLIST KeyPressed y CDATA #REQUIRED>
<!ATTLIST KeyPressed t CDATA #REQUIRED>
<!ATTLIST KeyPressed id CDATA #REQUIRED>
<!ATTLIST KeyPressed char CDATA #REQUIRED>

<!ELEMENT KeyGlass EMPTY>
```

```
<!ATTLIST KeyGlass x CDATA #REQUIRED>
<!ATTLIST KeyGlass y CDATA #REQUIRED>
<!ATTLIST KeyGlass t CDATA #REQUIRED>
<!ATTLIST KeyGlass char CDATA #REQUIRED>

<!ELEMENT KeyGlassInit EMPTY>
<!ATTLIST KeyGlassInit t CDATA #REQUIRED>
<!ATTLIST KeyGlassInit id CDATA #REQUIRED>

<!ELEMENT MotACopier EMPTY>
<!ATTLIST MotACopier mot CDATA #REQUIRED>
<!ATTLIST MotACopier t CDATA #REQUIRED>
```

B

Outils de la plate-forme E-Assiste

Nous présentons dans cette annexe les différents modules qui ont été réalisés pour la plate-forme E-Assiste.

B.7 Outils pour la partie sujet

B.7.1 Interpréteur de fichier

Cet outil permet de générer un clavier à partir d'un fichier de description du clavier. Ce fichier sous formalisme XML décrit la position, la taille et les diverses caractéristiques d'une touche ainsi que du ou des caractères qui sont associés à chaque touche. Ce fichier définit aussi le comportement de chaque touche lorsque le pointeur du dispositif de pointage survole ou sélectionne celle-ci. La définition (DTD) de ce langage balisé est donnée à la section B.9. La figure 59 montre le clavier généré à partir d'un fichier de ce type.

FIG. 59 – Exemple de clavier généré à partir d'un fichier XML

Exemple 7 (Description de la touche associée au caractère 'a')

```
<Touche>
   <Information visible="a" visibleShift="A" visibleAlt=""
virtualkey="65" keyevent="65"/>
   <Etat type="normal"><Position x="40" y="53" longueur="$longueurTouchePress"
 hauteur="$hauteurToucheNormal"/></Etat>
   <Etat type="select"><Position x="40" y="53" longueur="$longueurTouchePress"
hauteur="$hauteurToucheSelect"/></Etat>
   <Etat type="press"><Position  x="41" y="54" longueur="$longueurTouchePress"
hauteur="$hauteurTouchePress"/></Etat>
</Touche>
```

B.7.2 Interface de recopie de mots

Pour permettre les exercices de recopie de mot ou de phrase courte, nous avons réalisé le module présenté à la figure 60. Il présente sur la zone du haut le mot (ou séquence de mots) à saisir, et sur celle du bas ce qui a été saisi par le sujet. En cas d'erreur de saisie, le caractère erroné n'est pas affiché et un feedback sonore et visuel signale au sujet qu'il a fait une erreur. Celui-ci doit alors saisir à nouveau le caractère jusqu'à ce qu'il soit correctement saisi. Ce module de présentation est abonné aux messages transmettant les caractères sur le bus Ivy. Quand la chaîne de caractères a été intégralement saisie par le sujet, le module renvoie à son tour un message pour annoncer la fin de la séquence à saisir.

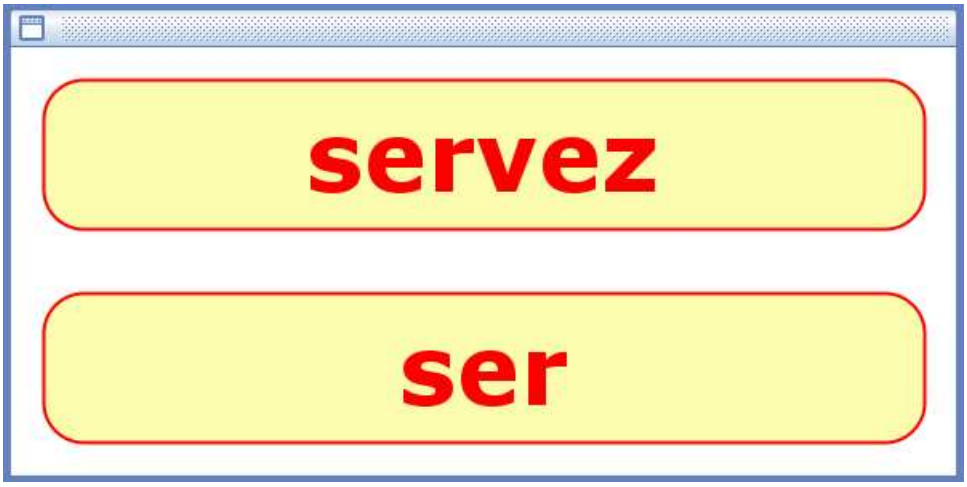


Fig. 60 – Interface pour les exercices de recopie de mot.

B.7.3 Bandeau de défilement de texte

Ce bandeau (cf. Figure 61) à été réalisé pour la recopie de texte long. Le but est de ne présenter au sujet que les 4 ou 5 prochains caractères qu'il doit saisir. Le caractère à saisir immédiatement est lui mis en gras et toujours à gauche du bandeau. Lorsqu'un sujet saisit un caractère :

- Si le caractère est correctement saisi, celui-ci disparaît du bandeau, les autres caractères sont alors décalés sur la gauche et le caractère suivant cette chaîne de caractères apparaît à droite du bandeau. Le nouveau caractère à saisir est mis en gras à gauche ;
- Si le caractère est erroné, la chaîne de caractères affichée sur le bandeau ne bouge pas, et un feedback sonore est enclenché pour prévenir l'utilisateur qu'il a saisi un mauvais caractère.

FIG. 61 – Bandeau de défilement du texte à saisir.

B.8 Outils pour la partie sauvegarde et analyse

B.8.1 Gestionnaire des traces temps réel

C'est cette interface (Figure 62) qui est affichée pour suivre en temps réel l'évolution des expérimentations en cours. Elle affiche en haut à gauche l'ensemble des expérimentations qui sont en cours, et sur la partie de droite la liste des données collectées de l'expérimentation sélectionnée.

C'est à partir de cette dernière qu'il est possible de faire une première analyse des résultats avec les boutons liés aux analyses de base en bas de l'interface. Pour pouvoir activer l'une ou l'autre des analyses, il faut que l'expérimentation sélectionnée soit terminée.

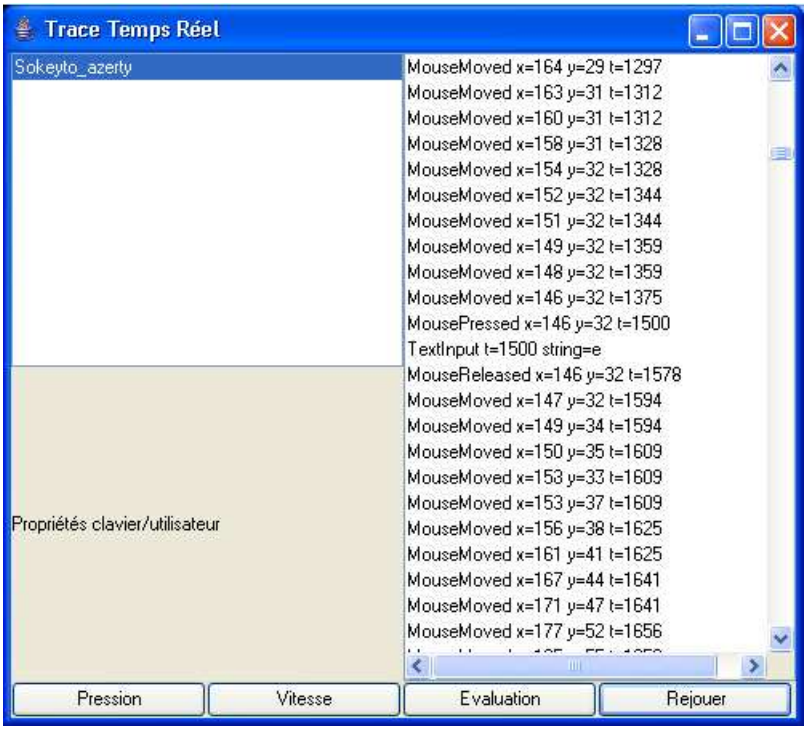


FIG. 62 – Interface de recueil des traces des diverses expérimentations en cours.

B.8.2 Analyseur des principales variables

A partir d'un fichier de données recueillies, il calcule un ensemble de métriques de base telles que la distance parcourue, le temps mis pour effectuer la tâche de saisie ou encore le taux d'erreur de saisie. Il les affiche dans une interface (cf. Figure 63) ainsi que le texte qui a été saisi (en bleu les caractères correctement saisis, et en rouge les caractères erronés).

B.8.3 Affichage de courbes et histogramme

A partir de données d'une expérimentation, les interfaces présentées respectivement dans les figures 64 et 65 donnent une représentation plus visuelle des données avec : la courbe représentant la vitesse du curseur du pointeur du dispositif de pointage tout au long de la tâche de saisie ; la durée de chaque pression/relâchement sur un bouton du dispositif de pointage, ou le temps de pression d'un stylet sur un écran tactile par exemple.

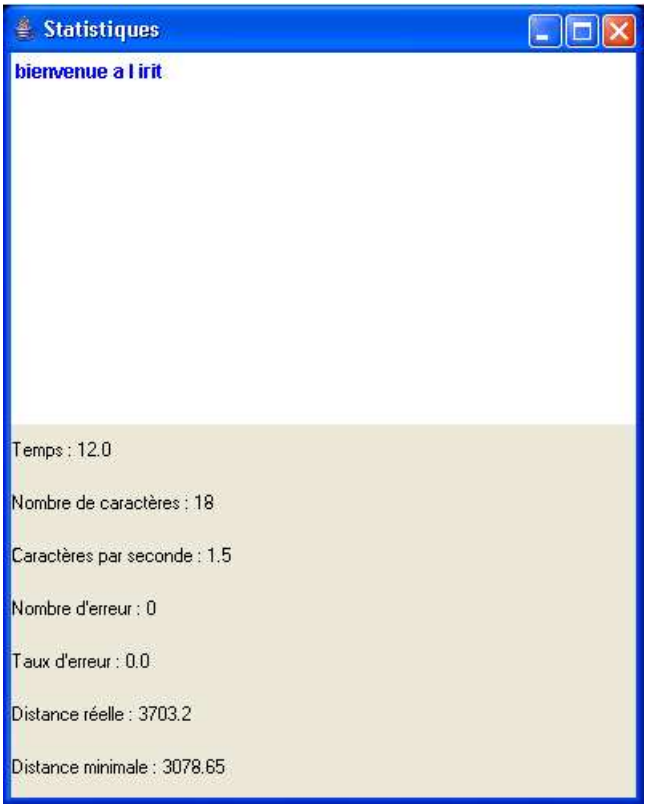


FIG. 63 – Interface de présentation des principaux résultats d'analyse

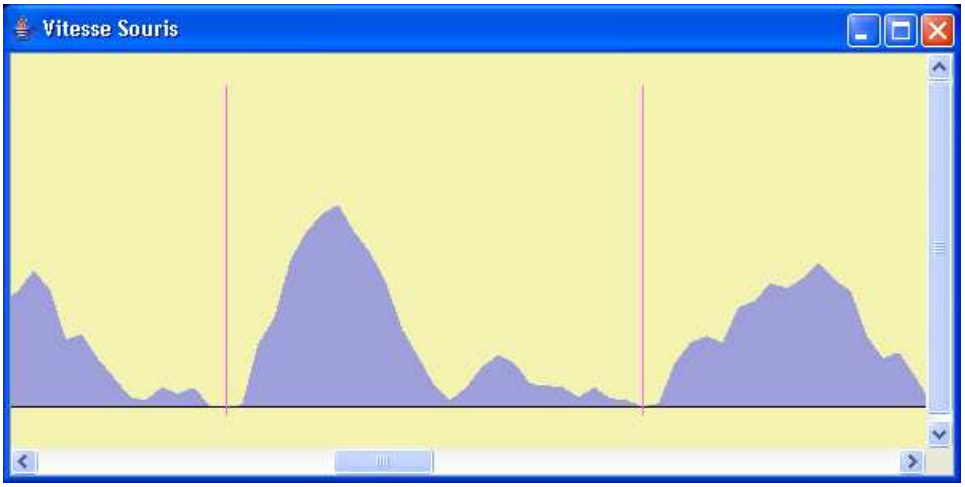


FIG. 64 – Courbe de vitesse du pointeur lors de la saisie.

B.8.4 Rejeu

Cette interface (cf Figure 66) permet de rejouer à l'identique la tâche de saisie que vient d'effectuer un sujet. Cela permet par exemple lors d'expérimentations à distance

FIG. 65 – Histogramme représentant les temps de pression sur les touches.

de voir le comportement du sujet à travers la trajectoire du pointeur de son dispositif de pointage.

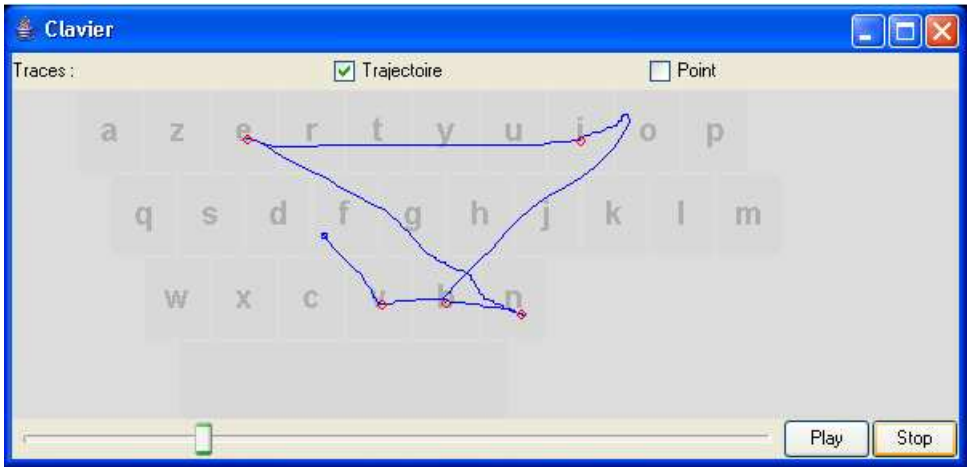


FIG. 66 – Interface de rejeu.

B.9 Document de définition pour le format de description des claviers logiciels

```
<!ELEMENT Clavier (Constante*,Image*,Definition*,Touche*,KeyGlass)>
<!ATTLIST Clavier x CDATA #REQUIRED>
<!ATTLIST Clavier y CDATA #REQUIRED>
```

```
<!ATTLIST Clavier longueur CDATA #REQUIRED>
<!ATTLIST Clavier hauteur CDATA #REQUIRED>
<!ATTLIST Clavier couleur CDATA #REQUIRED>

<!ELEMENT Constante EMPTY>
<!ATTLIST Constante nom CDATA #REQUIRED>
<!ATTLIST Constante valeur CDATA #REQUIRED>

<!ELEMENT Image (Ligne*,Triangle*,Rectangle*,Pixel*)>
<!ATTLIST Image nom CDATA #REQUIRED>

<!ELEMENT Ligne EMPTY>
<!ATTLIST Ligne couleur CDATA #FIXED "black">
<!ATTLIST Ligne epaisseur CDATA #REQUIRED>
<!ATTLIST Ligne x1 CDATA #REQUIRED>
<!ATTLIST Ligne y1 CDATA #REQUIRED>
<!ATTLIST Ligne x2 CDATA #REQUIRED>
<!ATTLIST Ligne y2 CDATA #REQUIRED>

<!ELEMENT Triangle EMPTY>
<!ATTLIST Triangle couleur CDATA #REQUIRED>
<!ATTLIST Triangle epaisseur CDATA #REQUIRED>
<!ATTLIST Triangle plein (true|false) #FIXED "true">
<!ATTLIST Triangle x1 CDATA #REQUIRED>
<!ATTLIST Triangle y1 CDATA #REQUIRED>
<!ATTLIST Triangle x2 CDATA #REQUIRED>
<!ATTLIST Triangle y2 CDATA #REQUIRED>
<!ATTLIST Triangle x3 CDATA #REQUIRED>
<!ATTLIST Triangle y3 CDATA #REQUIRED>

<!ELEMENT Rectangle EMPTY>
<!ATTLIST Rectangle couleur CDATA #REQUIRED>
<!ATTLIST Rectangle epaisseur CDATA #REQUIRED>
<!ATTLIST Rectangle plein (true|false) #FIXED "false">
<!ATTLIST Rectangle x CDATA #REQUIRED>
<!ATTLIST Rectangle y CDATA #REQUIRED>
```

```
<!ATTLIST Rectangle longueur CDATA #REQUIRED>
<!ATTLIST Rectangle hauteur CDATA #REQUIRED>

<!ELEMENT Pixel EMPTY>
<!ATTLIST Pixel couleur CDATA #REQUIRED>
<!ATTLIST Pixel x CDATA #REQUIRED>
<!ATTLIST Pixel y CDATA #REQUIRED>



<!ELEMENT Definition (Etat*)>

<!ELEMENT Touche (Information,Etat,Etat,Etat)>

<!ELEMENT Information EMPTY>
<!ATTLIST Information visible CDATA #REQUIRED>
<!ATTLIST Information visibleShift CDATA #IMPLIED>
<!ATTLIST Information visibleAlt CDATA #IMPLIED>
<!ATTLIST Information virtualkey CDATA #REQUIRED>
<!ATTLIST Information keyevent CDATA #REQUIRED>

<!ELEMENT Etat (Caracteristique*,Position*,Contour*,Fond*,Fonte*)>
<!ATTLIST Etat type (normal|select|press) #REQUIRED>

<!ELEMENT Caracteristique EMPTY>
<!ATTLIST Caracteristique type (rectangle|cercle|round-rectangle) #FIXED "rectangle">

<!ELEMENT Position EMPTY>
<!ATTLIST Position x CDATA #REQUIRED>
<!ATTLIST Position y CDATA #REQUIRED>
<!ATTLIST Position longueur CDATA #REQUIRED>
<!ATTLIST Position hauteur CDATA #REQUIRED>

<!ELEMENT Contour EMPTY>
<!ATTLIST Contour visible (true|false) #FIXED "true">
<!ATTLIST Contour epaisseur CDATA #REQUIRED>
<!ATTLIST Contour couleur CDATA #REQUIRED>
```

```
<!ELEMENT Fond EMPTY>
<!ATTLIST Fond couleur CDATA #REQUIRED>

<!ELEMENT Fonte EMPTY>
<!ATTLIST Fonte nom CDATA #REQUIRED>
<!ATTLIST Fonte type (plain|bold|italic|bold-italic) #FIXED "plain">
<!ATTLIST Fonte taille CDATA #REQUIRED>
<!ATTLIST Fonte couleur CDATA #REQUIRED>
```

Bibliographie

[Accot 01] J. Accot. *Les Tâches Trajectorielles en Interaction Homme-Machine : Cas des Tâches de Navigation.* PhD thesis, Université de Toulouse I, janvier 2001.

[Arnott 92] J.L. Arnott and M.Y. Javed. *Probabilistic character disambiguation for reduced keyboard using small text samples. Augmentative Alternative Communication*, 8 :215–223, 1992.

[Balakrishnan 04] R. Balakrishnan. *"Beating" Fitts' law : virtual enhancements for pointing facilitation. International Journal of Human-Computer Studies*, 61(6) :857–874, 2004.

[Baudisch 03] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson and A. Zierlinger. *Drag-and-Pop and Drag-and-Pick : Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems.* In Interact 2003, pages 57–64, Zurich, Suisse, août 2003.

[Beck 04] C. Beck, Seisenbacher G., Edelmayer G. and Zagler W.L.. *First User Test Results with the Predictive Typing System FASTY.* In K. Miesenberger, J. Klaus, W. Zagler. and D. Burger, editors, 9th ICCHP : Computer Helping People with Special Needs, pages 813–819. Lecture Notes in Computer Science, LCNS 3118, 2004.

[Bellman 98] T. Bellman and I.S. MacKenzie. *A probabilistic character layout strategy for mobile text entry.* In Proceedings of Graphics Interface, pages 168 – 176, Toronto, 1998. Canadian Information Processing Society.

[Bier 93] E.A. Bier, M.C. Stone, K. Pier, W. Buxton and T.D. DeRose. *Toolglass and magic lenses : the see-through interface.* In SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 73–80, New York, NY, USA, 1993. ACM Press.

[Blanch 04] R. Blanch, Y. Guiard and M. Beaudouin-Lafon. *Semantic pointing : improving target acquisition with control-display ratio adaptation.* In CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 519–526, New York, NY, USA, 2004. ACM Press.

[Boissière 90] Ph. BOISSIÈRE. VITIPI *un système auto-organisationnel pour faciliter le dialogue écrit homme-machine*. PhD thesis, Université Paul Sabatier, 1990.

[Boissière 06] Ph. BOISSIÈRE. *An overview of Writing Assistance Systems : VITIPI as an example of a Writing Assistance System. Universal Access in the Information Society*, à paraître, 2006.

[Buisson 02] M. BUISSON, A. BUSTICO, St. CHATTY, F.R. COLIN, Y. JESTIN, S. MAURY, Ch. MERTZ and Ph. TRUILLET. *Ivy : un bus logiciel au service du développement de prototypes de systèmes interactifs*. In 14^{th} Conférence Francophone sur l'Interaction Homme-Machine (IHM'02), pages 223–226, New York, NY, USA, 2002. ACM Press.

[Buzing 03] P. BUZING. Comparing different keyboard layouts : Aspects of qwerty, dvorak and alphabetical keyboards. Student papers : Human-Computer interaction course, 2003.

[Bäck 93] Th. BÄCK, G. RUDOLPH and H.P. SCHWEFEL. *Evolutionary Programming and Evolution Strategies : Similarities and Differences*. In D.B. FOGEL and W. ATMAR, editors, Second Annual Conference on Evolutionary Programming, pages 11–22. Evolutionary Programming Society, San Diego CA, 1993.

[Bérard 04a] Ch. BÉRARD. *Clavier-écran : concevoir avec les utilisateurs*. In Handicap 2004, pages 83–88, 2004.

[Bérard 04b] Ch. BÉRARD and D. NIEMEIJER. *Evaluating effort reduction through different word prediction systems*. In IEEE SMC 2004 :International Conference on Systems, Man and Cybernetics, pages 2658–2663, 2004.

[Callahan 88] J. CALLAHAN, D. HOPKINS, M. WEISER and B. SHNEIDERMAN. *An empirical comparison of pie vs. linear menus*. In CHI '88 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 95–100, New York, NY, USA, 1988. ACM Press.

[Cantegrit 01] B. CANTEGRIT and J.M. TOULOTTE. *Réflexions sur l'aide à la communication des personnes présentant un handicap moteur*. In TALN 01, pages 193–202, Tours, 2-5 juillet 2001.

[Card 78] S.K. CARD, W. ENGLISH and B.J. BURR. *Evaluation of mouse, rate-controlled isometric joystick,step keys, and text keys for text selection on a CRT. Ergonomics*, 21 :601–613, 1978.

[Card 80] S.K. CARD, T.P. MORAN and A. NEWELL. *The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of the ACM*, 23(7) :396–410, July 1980.

[Card 83] S.K. CARD, T.P. MORAN and A. NEWELL. *The psychology of human-computer interaction. Hillsdale, NJ : Lawrence Erlbaum Associates, Inc*, 1983.

[Caux 95] C. CAUX, H. PIERREVAL and M.C. PORTMAN. *Les Algorithmes Génétiques et Leur Application d'Ordonnancement. Automatique, Productique, Informatique Industrielle*, 29(4-5) :409–443, 1995.

[Chubon 88] R.A. Chubon and M.R. Hester. *An enhanced standard computer keyboard system for single-finger and typing-stick typing. Rehabilitation Research and Development*, 25(4) :17–24, 1988.

[Cooper 83] W.E. Cooper. Cognitive aspects of skilled typing. New York : Springer-Verlag, 1983.

[Cormen 94] T. Cormen, Ch. Leiserson and R. Rivest. Introduction à l'algorithmique. Dunod, 1994.

[Darragh 90] J.J. Darragh, I.H. Witten and M.L. James. *The Reactive Keyboard : A Predictive Typing Aid. Computer*, 23(11) :41–49, 1990.

[Darwin 59] C. Darwin. *On the Origin of Species by means of natural selection, or the Preservations of favored races in the struggle of life.* 1859.

[De Calmès 98] M. De Calmès and G. Pérennou. *BDLEX : a Lexicon for Spoken and Written French.* In 1st International Conference on Langage Resources & Evaluation , Grenade, pages 1129–1136, Paris, 28-30 mai 1998. A.Rubio, N. Gallardo, R. Castro, A. Tejada, ELRA.

[De Jong 57] J.R. De Jong. *The effects of increasing skill on cycle time and its consequences for time standards. Ergonomics*, 6 :51–60, 1957.

[Douglas 97] S. Douglas and A.K. Mithal. *The Ergonomics of Computer Pointing Devices. Springer, Berlin*, 1997.

[Dréo 03] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard. Métaheuristiques pour l'optimisation difficile. Eyrolles, 2003.

[Dunlop 00] M.D. Dunlop and A. Crossan. *Predictive Text Entry Methods for Mobile Phones. Personal Technologies*, 4(2) :134–143, 2000.

[Dvorak 36] A. Dvorak, N.L. Merrick, W.L. Dealey and G.C. Ford. Typewriting behavior. American Book Co., New York, USA, 1936.

[Eggers 03] J. Eggers, D. Feillet, S. Kehl, M.O. Wagner and B. Yannou. *Optimization of the keyboard arrangement problem using an Ant Colony algorithm. European Journal of Operational Research*, 148 :672–686, 2003.

[Evreinova 04] T. Evreinova, G. Evreinov and R. Raisamo. *Four-Key Text Entry for Physically Challenged People.* 2004.

[Fitts 54] P.M. Fitts. *The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology*, 47 :381–391, 1954.

[Fitts 64] P.M. Fitts and J.R. Peterson. *Information capacity of discrete motor response. Journal of Experimental Psychology*, 67 :103–112, 1964.

[Fogel 66] L.J. Fogel, A.J. Owens and M.J. Walsh. Artificial intelligence through simulated evolution. Wiley, New York, NY, 1966.

[Gentner 83] D. R. Gentner, J. T. Grudin, S. Larochelle, D. A. Norman and D. E. Rumelhart. Cognitive aspects of skilled typing, Chapitre A glossary of terms including a classification of typing errors, pages 39–44. New York : Springer-Verlag, 1983.

[Getschow 86] C.O. GETSCHOW, M.J. ROSEN and C. GOODENOUGH-TREPAGNIER. *A systematic approach to design of a minimum distance alphabetical keyboard.* In Proceedings of Rehabilitation Engineering Society of North America - RESNA 9^{th} Annual Conference, pages 396–398, 1986.

[Gillan 90] D.J. GILLAN, K. HOLDEN, S. ADAM, M. RUDISILL and L. MAGEE. *How does Fitts' law fit pointing and dragging ?* In CHI '90 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 227–234, New York, NY, USA, 1990. ACM Press.

[Glaser 81] R.E. GLASER. *A telephone communication aid for the deaf.* In Johns Hopkins First Nat. Search for Applications of Personal Computers to Aid the Handiccaped, pages 11–15, 1981.

[Goldberg 89] D.E. GOLDBERG. *Genetic Algorithms in Search. Optimisation and Machine Learning*, 1989.

[Gong 05] J. GONG, B. HAGGERTY and P. TARASEWICH. *An enhanced multitap text entry method with predictive next-letter highlighting.* In CHI '05 : CHI '05 extended abstracts on Human factors in computing systems, pages 1399–1402, New York, NY, USA, 2005. ACM Press.

[Green 04] N. GREEN, J. KRUGER, C. FALDU and R.St. AMANT. *A reduced QWERTY keyboard for mobile text entry.* In CHI '04 : CHI '04 extended abstracts on Human factors in computing systems, pages 1429–1432, New York, NY, USA, 2004. ACM Press.

[Grefenstette 87] J.J. GREFENSTETTE. *Genetic Algorithms and Their Applications.* In J.J. GREFENSTETTE, editor, Proceedings of the second international conference on genetic algorithms and their applications, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[Grossman 05] T. GROSSMAN and R. BALAKRISHNAN. *The bubble cursor : enhancing target acquisition by dynamic resizing of the cursor's activation area.* In CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 281–290, New York, NY, USA, 2005. ACM Press.

[Grover 98] D.L. GROVER, M.T. KING and C.A. KUSCHLER. Reduced keyboard disambiguating computer. Seattle, WA : Tegic Communications Incorporated, 1998.

[Guiard 04] Y. GUIARD and M. BEAUDOUIN-LAFON. *Preface : Fitts' law 50 years later : Applications and contributions from human-computer interaction. International Journal of Human-Computer Studies*, 61(6) :747–750, 2004.

[Hansen 04] J.P. HANSEN, K. TORNING, A.S. JOHANSEN, K. ITOH and A. AOKI. *Gaze typing compared with input by head and hand.* In ETRA'2004 : Proceedings of the Eye tracking research & applications symposium on Eye tracking research & applications, pages 131–138, New York, NY, USA, 2004. ACM Press.

[Hick 52] W.E. HICK. *On the rate of gain of information. Quarterly Journal of Experimental Psychology*, 4 :11–36, 1952.

[Holland 75] J.H. HOLLAND. Adaptation in natural and artificial systems. University of Michigan Press, 1975.

[Hughes 02] D. HUGHES, J. WARREN and O. BUYUKKOKTEN. *Empirical Bi-Action Tables : A Tool for the Evaluation and Optimization of Text-Input Systems. Application I : Stylus Keyboards. Human-Computer Interaction*, 17 :131–168, 2002.

[Hunter 00] M. HUNTER, S. ZHAI and B.A. SMITH. *Physics-based graphical keyboard design.* In CHI '00 : CHI '00 extended abstracts on Human factors in computing systems, pages 157–158. ACM Press, 2000.

[Hyman 53] R. HYMAN. *Stimulus information as a determinant of reaction time. Journal of Experimental Psychology*, 45 :188–196, 1953.

[Ingmarsson 04] M. INGMARSSON, D. DINKA and S. ZHAI. *TNT : a numeric keypad based text input method.* In CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 639–646, New York, NY, USA, 2004. ACM Press.

[Isokoski 00] P. ISOKOSKI and R. RAISAMO. *Device Independent Text Input : A Rationale and an Example.* In Proceedings of the Working Conference on Advanced Visual Interfaces AVI2000, pages 76–83, Palermo, Italy, 2000. ACM.

[Isokoski 04a] P. ISOKOSKI. *Manual Text Input : Experiments, Models, and Systems.* PhD thesis, University of Tampere, 2004.

[Isokoski 04b] P. ISOKOSKI. *Performance of menu-augmented soft keyboards.* In CHI '04 : Proceedings of the 2004 conference on Human factors in computing systems, pages 423–430. ACM Press, 2004.

[Isokoski 04c] P. ISOKOSKI and R. RAISAMO. *Quikwriting as a multi-device text entry method.* In NordiCHI '04 : Proceedings of the third Nordic conference on Human-computer interaction, pages 105–108. ACM Press, 2004.

[Jagacinski 85] R. J. JAGACINSKI and D. L. MONK. *Fitts' Law in Two Dimensions with Hand and Head Movements. Journal of Motor Behavior*, 17(1) :77–95, 1985.

[Johnson 81] A.B. JOHNSON and R.F. HAGSTAD. *DTMF telecommunications for the deaf and speech impaired.* In Johns Hopkins First Nat. Search for Applications of Personal Computers to Aid the Handiccaped, pages 29–32, 1981.

[Jong 75] K.A. De JONG. *An analysis of the Behaviorof a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[Khuri 95] S. KHURI and M. SCHÜTZ. *Evolutionary Heuristics for the Bin Packing Problem.* In ICANNGA'95, 1995.

[King 95] M.T. KING, C.A. KUSHLER and D.A. GROVER. *JustType - Efficient Communication with Eight Keys.* In RESNA'95, pages 94–96, 1995.

[Koester 94] H.H. KOESTER and S.P. LEVINE. *Validation of a Keystroke-level Model for a Text Entry System Used by People with Disabilities.* In Proceedings of ASSETS '94 : 1st Annual ACM Conference on Assistive Technologies, pages 115–122. CA. New York :ACM, 1994.

[Kristensson 04] P.-O. KRISTENSSON and S. ZHAI. *SHARK2 : a large vocabulary shorthand writing system for pen-based computers.* In UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology, pages 43–52. ACM Press, 2004.

[Kruskal 83] J. B. KRUSKAL. *An Overview of Sequence Comparison.* In David SANKOFF and Joseph B. KRUSKAL, editors, Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison, page382. Addison-Wesley, Reading, MA, 1983.

[Kurtenbach 93] G. KURTENBACH, A. SELLEN and W. BUXTON. *An empirical evaluation of some articulatory and cognitive aspects of "marking menus". Journal of Human Computer Interaction*, 8(1) :1–23, 1993.

[Landauer 85] T. K. LANDAUER and D. W. NACHBAR. *Selection from alphabetic and numeric menu trees using a touch screen : breadth, depth, and width.* In CHI '85 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 73–78, New York, NY, USA, 1985. ACM Press.

[Lesher 98] G.W. LESHER, B.J. MOULTON and D.J. HIGGINBOTHAM. *Optimal Character Arrangements for Ambiguous Keyboards. IEEE Transactions on Rehabilitation Engineering*, 6(4) :415–423, decembre 1998.

[Lesher 00] G.W. LESHER and B.J. MOULTON. *A method for optimizing single-finger keyboards.* In Proceedings of Rehabilitation Engineering Society of North America - RESNA Annual Conference, pages 91–93, 2000.

[Levenshtein 66] V. I. LEVENSHTEIN. *Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics-Doklady*, 10 :707–710, 1966.

[Levine 87] S.H. LEVINE, C. GOODENOUGHT-TREPAGNIER, C.O. GETSCHOW and S.L. MINNEMAN. *Multi-character key text entry using computer disambiguation.* In RESNA, pages 177–179, Washington, 1987.

[Levine 90] S.H. LEVINE and C. GOUDENOUGH-TREPAGNIER. *Customised text entry devices for motor-impaired users. Applied Ergonomics*, 21 :55–62, 1990.

[Lewis 99a] J.R. LEWIS, M.J. LALOMIA and P.J. KENNEDY. *Development of a digram-based typing key layout for single-finger/stylus input.* In Proceedings of the Human Factors and Ergonomics Society 43^{rd} Annual Meeting - HFES'99, pages 415–419, Santa Monica, 1999. CA : Human Factors and Ergonomics Society.

[Lewis 99b] J.R. LEWIS, M.J. LALOMIA and P.J. KENNEDY. *Evaluation of typing key layouts for stylus input.* In Proceedings of the Human Factors and Ergonomics Society 43^{rd} Annual Meeting - HFES'99, pages 420–424, Santa Monica, 1999. CA : Human Factors and Ergonomics Society.

[Lin 65] S. Lin. *Computer solutions of the traveling salesman problem. Bell System Technical Journal*, 44 :2245–2269, 1965.

[Lin 73] S. Lin and B.W. Kerninghan. *An effective heuristic algorithm for the traveling salesman problem. Operations Research*, 21 :498–516, 1973.

[Lin 93] F.T. Lin, C.Y. Kao and C.C. Hsu. *Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems. IEEE Transactions on Systems, Man and Cybernetic*, 23(6) :1752–1767, 1993.

[LoPresti 00] E. LoPresti, D.M. Brienza, J. Angelo, L. Gilbertson and J. Sakai. *Neck range of motion and use of computer head controls.* In Assets '00 : Proceedings of the fourth international ACM conference on Assistive technologies, pages 121–128, New York, NY, USA, 2000. ACM Press.

[LoPresti 03] E.F. LoPresti, D.M. Brienza, J. Angelo and L. Gilbertson. *Neck range of motion and use of computer head controls. Journal of Rehabilitation Research and Development*, 40(3) :199–212, mai/juin 2003.

[MacKenzie 91a] I.S. MacKenzie. *Fitts' law as a performance model in human-computer interaction.* PhD thesis, University of Toronto, 1991.

[MacKenzie 91b] I.S. MacKenzie, A. Sellen and W. Buxton. *A comparison of input devices in element pointing and dragging tasks.* In CHI '91 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 161–166. ACM Press, 1991.

[MacKenzie 92a] I.S. MacKenzie. *Fitts' law as a research and design tool in human-computer interaction. Human-Computer Interaction*, 7 :91–139, 1992.

[MacKenzie 92b] I.S. MacKenzie and W. Buxton. *Extending Fitts' law to two-dimensional tasks.* In CHI '92 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 219–226. ACM Press, 1992.

[MacKenzie 94] I.S. MacKenzie, B. Nonnecke, S. Riddersma, C. McQueen and M. Meltz. *Alphanumeric entry on pen-based computers. International Journal of Human-Computer Studies*, 41 :775 – 792, 1994.

[MacKenzie 99a] I.S. MacKenzie and S.X. Zhang. *The design and evaluation of a high-performance soft keyboard.* In CHI '99 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 25–31. ACM Press, 1999.

[MacKenzie 99b] I.S. MacKenzie, S.X. Zhang and R.W. Soukoreff. *Text entry using soft keyboards. Behavior and Information Technology*, 18(4) :235–244, 1999.

[MacKenzie 01a] I.S. MacKenzie, H. Kober, D. Smith, T. Jones and E. Skepner. *LetterWise : prefix-based disambiguation for mobile text input.* In UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology, pages 111–120. ACM Press, 2001.

[MacKenzie 01b] I.S. MacKenzie and S.X. Zhang. *An empirical investigation of the novice experience with soft keyboards. Behaviour and Information Technology*, 20 :411–418, 2001.

[MacKenzie 02a] I.S. MacKenzie. *KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques.* In Mobile HCI '02 : Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, pages 195–210. Springer-Verlag, 2002.

[MacKenzie 02b] I.S. MacKenzie. *Mobile text entry using three keys.* In NordiCHI '02 : Proceedings of the second Nordic conference on Human-computer interaction, pages 27–34, New York, NY, USA, 2002. ACM Press.

[MacKenzie 02c] I.S. MacKenzie and R.W. Soukoreff. *A character-level error analysis technique for evaluating text entry methods.* In NordiCHI '02 : Proceedings of the second Nordic conference on Human-computer interaction, pages 243–246. ACM Press, 2002.

[MacKenzie 02d] I.S. MacKenzie and R.W. Soukoreff. *Text entry for mobile computing : Models and methods, theory and practice. Human-Computer Interaction*, 17 :147–198, 2002.

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

[Magnien 04] J.L. Magnien, L.and Bouraoui and N. Vigouroux. *Mobile devices : soft keyboard text-entry enhanced by Visual Cues.* In Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing, pages 158–165, New York, NY, USA, 2004. ACM Press.

[Magnien 05] L. Magnien. *Saisie de texte sur dispositifs nomades : Propositions et évaluations de solutions pour la saisie au stylet sur claviers logiciels.* PhD thesis, Université Paul Sabatier, Toulouse III, 2005.

[Majaranta 02] P. Majaranta and K.J. Räihä. *Twenty years of eye typing : systems and design issues.* In ETRA '02 : Proceedings of the symposium on Eye tracking research & applications, pages 15–22, New York, NY, USA, 2002. ACM Press.

[Masui 98] T. Masui. *An efficient text input method for pen-based computers.* In CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 328–335, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[Matias 93] E. Matias, I.S. MacKenzie and W. Buxton. *Half-QWERTY : a one-handed keyboard facilitating skill transfer from QWERTY.* In Proceedings fot the CHI 93, pages 88 – 94. ACM, 1993.

[Matias 96] E. Matias, I.S. MacKenzie and W. Buxton. *One-Handed Touch-Typing on a QWERTY Keyboard. Human-Computer Interaction*, 11 :1 – 27, 1996.

[Maurel 01] D. MAUREL, N. ROSSI and R. THIBAULT. *HandiAS : un système multilingue pour l'aide à la communication des personnes handicapées.* In TALN'01, volume 2, pages 203–212, 2001.

[Mayzner 65] M.S. MAYZNER and M.E. TRESSELT. *Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. Psychonomics Monograph Supplements*, 1(2) :13–32, 1965.

[McGuffin 02] M. MCGUFFIN and R. BALAKRISHNAN. *Acquisition of expanding targets.* In CHI '02 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 57–64, New York, NY, USA, 2002. ACM Press.

[McQueen 94] C. MCQUEEN, I.S. MACKENZIE, B. NONNECKE and S. RIDDERSMA. *A comparison of four methods of numeric entry on pen-based computers.* In Graphics Interface '94, pages 75–82, 1994.

[Metropolis 53] N. METROPOLIS, A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER and E. TELLER. *Equation of state calculations by fast computing machines. Journal of Chemical Physics*, 21 :1087–1090, 1953.

[Meyer 90] D.E MEYER, K.J.E SMITH, S. KORNBLUM, R.A. ABRAMS and C.E. WRIGHT. *Speed-accuracy tradeoffs in aimed movements : toward a theory of rapid voluntary action. Attention and Performance XIII*, pages 173–226, 1990.

[Ménier 01] G. MÉNIER and F. POIRIER. *Système adaptatif de prédiction de texte.* In TALN 01, pages 213–222, Tours, 2-5 juillet 2001.

[Niemeijer 05] D. NIEMEIJER. *In memoriam of Christian Bérard : Striving for effort reduction through on-screen keyboard word prediction.* In Workshop AAATE, 2005.

[Norman 82] D.A. NORMAN and D. FISHER. *Why alphabetic keyboards are not easy to use : Keyboard layout doesn't much matter. Human factor*, 24 :509–519, 1982.

[Oriola 05] B. ORIOLA, A. HERMET and M. RAYNAL. *User-Centered Design applied to Non Visual Interaction.* In 3rd Int. Conf. on Universal Access in Human-Computer Interaction (UAHCI 2005) , Las Vegas, USA, pageCDRom, ISBN 0-8058-5807-5, 22-27 juillet 2005. Lawrence Erlabaum Associates (LEA).

[Pavlovych 03] A. PAVLOVYCH and W. STUERZLINGER. *Less-Tap : A Fast and Easy-to-learn Text Input Technique for Phones.* In Graphics Interface 2003, pages 319–326. Canadian Information Processing Society, 2003.

[Pavlovych 04] A. PAVLOVYCH and W. STUERZLINGER. *Model for non-expert text entry speed on 12-button phone keypads.* In CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 351–358, New York, NY, USA, 2004. ACM Press.

[Pavlovych 05] A. PAVLOVYCH and W. STUERZLINGER. *An analysis of Novice Text Entry Performance on Large Interactive Wall Surfaces.* In HCI International 2005, pages CD–Rom. Lawrence Erlbaum, 2005.

[Perlin 98] K. PERLIN. *Quikwriting : continuous stylus-based text entry.* In Proceedings of the ACM UIST 98, pages 215–216. ACM, 1998.

[Plamondon 97] R. PLAMONDON and A.M. ALIMI. *Speed / accuracy trade-offs in target-directed movements. Behavior and Brain Sciences*, 20 :279–349, 1997.

[Poirier 04] F. POIRIER and I. SCHADLE. *Etat de l'art des méthodes de saisie de données sur dispositifs nomades. Typologie des approches.* In 16^{th} Conférence Francophone sur l'Interaction Homme-Machine (IHM'04), pages 133–140, 2004.

[Pévédic 97] B. Le PÉVÉDIC. *Prédiction morphosyntaxique évolutive dans un système d'aide à la saisie de textes pour des personnes handicapées physique.* PhD thesis, Ecole doctorale Sciences pour l'ingénieur de Nantes, 1997.

[Rau 94] H. RAU and S.S. SKIENA. *Dialing for Documents : An Experiment in Information Theory.* In Proc. of the 7th Annual Symposium on User Interface Software and Technology (UIST'94), pages 147–155, Marina del Rey, CA, 1994.

[Raynal 05] M. RAYNAL and N. VIGOUROUX. *Genetic algorithm to generate optimized soft keyboard.* In CHI '05 : CHI '05 extended abstracts on Human factors in computing systems, pages 1729–1732, New York, NY, USA, 2005. ACM Press.

[Rechenberg 73] I. RECHENBERG. Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart, 1973.

[Reeves 95] C.R. REEVES. *A Genetic Approach to Bin-Packing.* In INWGA, pages 35–49, Vaasa, janvier 1995.

[Schadle 02] I. SCHADLE, J.Y. ANTOINE, B. Le PÉVÉDIC and F. POIRIER. *SibyLettre : Prédiction de lettre pour la communication assistée. Revue d'Interaction Homme-Machine*, 3(2) :115–133, 2002.

[Schadle 03] I. SCHADLE. SIBYLLE *: système linguistique d'aide à la communication pour les personnes handicapées.* PhD thesis, Université de Bretagne Sud, 2003.

[Schadle 04] I. SCHADLE and F. POIRIER. *Sibylle : un système d'aide à la saisie de texte.* In 16^{th} Conférence Francophone sur l'Interaction Homme-Machine (IHM'04), pages 141–146, 2004.

[Schwefel 77] H.P. SCHWEFEL. Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie, volume 26 of *Interdisciplinary Systems Research.* Birkhäuser, Basle, 1977.

[Shannon 49] C.E. SHANNON and W. WEAVER. *The mathematical theory of communications. University of Illinois Press*, 1949.

[Silfverberg 00] M. SILFVERBERG, I.S. MACKENZIE and P. KORHONEN. *Predicting text entry speed on mobile phones.* In CHI '00 : Proceedings of the

SIGCHI conference on Human factors in computing systems, pages 9–16. ACM Press, 2000.

[Sirisena 02] A. SIRISENA. Mobile text entry. Technical Report, University of Canterbury, 2002.

[Smith 01] B.A. SMITH and S. ZHAI. *Optimised virtual keyboards with and without alphabetical ordering : A novice user study.* In Proceedings of Interact 2001 - IFIP International Conference on Human-Computer Interaction, pages 92–99, Tokyo, Japon, 2001.

[Soukoreff 95] R.W. SOUKOREFF and I.S. MACKENZIE. *Theoretical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard. Behavior and Information Technology*, 14(6) :370–379, 1995.

[Soukoreff 01] R.W. SOUKOREFF and I.S. MACKENZIE. *Measuring errors in text entry tasks : an application of the Levenshtein string distance statistic.* In CHI '01 : CHI '01 extended abstracts on Human factors in computing systems, pages 319–320. ACM Press, 2001.

[Soukoreff 02] R.W. SOUKOREFF. TEXT ENTRY FOR MOBILE SYSTEMS *: Models, Measures, and Analyses for Text Entry Research.* PhD thesis, York University, 2002.

[Soukoreff 03a] R.W. SOUKOREFF and I.S. MACKENZIE. *Input-based language modelling in the design of high performance text input techniques.* In Proceedings of Graphics Interface, pages 89–96, 2003.

[Soukoreff 03b] R.W. SOUKOREFF and I.S. MACKENZIE. *Metrics for text entry research : an evaluation of MSD and KSPC, and a new unified error metric.* In CHI '03 : Proceedings of the conference on Human factors in computing systems, pages 113–120. ACM Press, 2003.

[Soukoreff 04] R.W. SOUKOREFF and I.S. MACKENZIE. *Recent developments in text-entry error rate measurement.* In CHI '04 : Extended abstracts of the 2004 conference on Human factors and computing systems, pages 1425–1428. ACM Press, 2004.

[Sugimoto 96] M. SUGIMOTO and K. TAKAHASHI. *SHK : single hand key card for mobile devices.* In CHI '96 : Conference companion on Human factors in computing systems, pages 7–8, New York, NY, USA, 1996. ACM Press.

[Uguen 05] G. UGUEN and F. POIRIER. *Saisie de données pour interfaces réduites avec Glyph : principes, niveaux de saisie et évaluations théoriques.* In 17^{th} Conférence Francophone sur l'Interaction Homme-Machine (IHM'05), pages 11–18, Toulouse, 27-30 septembre 2005.

[Vella 05] F. VELLA, N. VIGOUROUX and Ph. TRUILLET. *SOKEYTO : a design and simulation environment of software keyboards.* In A PRUSKI and H KNOPS, editors, Assistive technology : from virtuality to reality - 8th European conference for the advancement of assistive technology in europe (AAATE 2005) , Lille, France, pages 723–727, ISBN 1-58603-543-6, 6-9 septembre 2005. IOS Press.

[Venolia 94] D. VENOLIA and F. NEIBERG. *T-Cube : a fast, self-disclosing pen-based alphabet.* In CHI '94 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 265–270, New York, NY, USA, 1994. ACM Press.

[Venolia 01] G.D. VENOLIA, J. GOODMAN, K. STEURY and C. PARKER. Language modeling for soft keyboards. Technical Report MSR-TR-2001-118, Microsoft Research, November 2001.

[Vigouroux 04] N. VIGOUROUX, F. VELLA, Ph. TRUILLET and M. RAYNAL. *Evaluation of AAC for Text Input by two Groups of Subjects : Able-bodies Subjects and Disabled Motor Subjects.* 2004.

[Ward 00] D.J. WARD, A.F. BLACKWELL and D.J.C. MACKAY. *Dasher - a data entry interface using continuous gestures and language models.* In UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology, pages 129–137, New York, NY, USA, 2000. ACM Press.

[Welford 68] A. T. WELFORD. Fundamentals of skill. Methuen, 1968.

[Wester 03] M. WESTER. User evaluation of a word prediction system. Master's thesis, Department of Linguistics, Uppsala University, 2003.

[Woodworth 99] R.S. WOODWORTH. *The accuracy of voluntary movement. Psychological review*, 3(2) :1–14, 1899.

[Yamada 80] H. YAMADA. *A historical study of typewriters and typing methods : from the position of planning Japanese parallels. Journal of Information Processing*, 2(4) :175–202, 1980.

[Zhai 00] S. ZHAI, M. HUNTER and B.A. SMITH. *The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design.* In UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology, pages 119–128. ACM Press, 2000.

[Zhai 01] S. ZHAI and B.A. SMITH. *Alphabetically biased virtual keyboards are easier to use : layout does matter.* In CHI '01 : CHI '01 extended abstracts on Human factors in computing systems, pages 321–322. ACM Press, 2001.

[Zhai 02a] S. ZHAI, B.A. SMITH and M. HUNTER. *Performance Optimization of Virtual Keyboards. Human-Computer Interaction*, 17(2&3) :229–270, 2002.

[Zhai 02b] S. ZHAI, A. SUE and J. ACCOT. *Movement model, hits distribution and learning in virtual keyboarding.* In CHI '02 : Proceedings of the SIGCHI conference on Human factors in computing systems, pages 17–24. ACM Press, 2002.

[Zhai 03a] S. ZHAI, S. CONVERSY, M. BEAUDOUIN-LAFON and Y. GUIARD. *Human on-line response to target expansion.* In CHI '03 : Proceedings of the conference on Human factors in computing systems, pages 177–184. ACM Press, 2003.

[Zhai 03b] S. Zhai and P.-O. Kristensson. *Shorthand writing on stylus keyboard.* In CHI '03 : Proceedings of the conference on Human factors in computing systems, pages 97–104. ACM Press, 2003.

[Zhang 98] S.X. Zhang. A high performance soft keyboard for mobile systems. Master's thesis, The University of Guelph, 1998.

Résumé

Depuis l'émergence du web dans les années 1990, l'informatique est devenue aujourd'hui un formidable vecteur de communication utilisable par le plus grand nombre. Les lois successives du gouvernement français autour de "l'administration électronique" (e-administration) permettent l'accessibilité et l'accès à de nombreux services, notamment pour les personnes handicapées.

Un des problèmes prégnants pour les personnes handicapées moteurs des membres supérieurs demeure le problème de la saisie de l'information sur ordinateur. L'utilisation de périphériques de pointage adaptés (trackball, eye-tacking, ...) associés à un clavier de type logiciel (affiché sur l'écran) permet, dans une certaine mesure, à ces personnes d'utiliser un ordinateur. Afin d'optimiser la saisie de texte pour ce type de population, des recherches sont menées depuis une vingtaine d'années. De plus, l'émergence des dispositifs mobiles dépourvus de clavier physique (PDA, Tablet PC, ...), a accentué la nécessité de ces recherches, pour les appliquer "à tous".

La problématique de cette thèse concerne l'optimisation et l'évaluation de nouveaux systèmes d'aide à la communication pour personnes handicapées moteur des membres supérieurs. Ces travaux s'articulent autour de deux propositions d'optimisation :

- d'une part, l'application d'une méta-heuristique d'optimisation connue (les algorithmes génétiques) pour optimiser la disposition des caractères sur le clavier logiciel et ainsi minimiser les déplacements à effectuer lors de saisie de texte en français. Le clavier GAG (Généré par Algorithme Génétique) est le résultat de cette optimisation.
- d'autre part, le système KeyGlass : celui-ci propose au fur et à mesure de la saisie des caractères supplémentaires à proximité du dernier caractère qui a été saisi. Ceci a pour but de diminuer les distances (avec le pointeur du dispositif de pointage) nécessaire à parcourir pour saisir du texte.

Pour pouvoir discuter de la validité des optimisations réalisées et de leur utilisabilité, il est nécessaire de définir une métrique commune afin de pouvoir évaluer et comparer les systèmes de saisie de texte. Devant le manque de ressources, de corpus de tests et d'outils standards d'évaluation, nous avons réalisé la plate-forme E-Assiste. Cette plate-forme d'évaluation répond à des objectifs d'étude des usages de nos systèmes en situation d'interaction.

Nous avons pu ainsi montrer que nos systèmes apportaient un gain significatif en vitesse de saisie de texte et diminution des distances à parcourir par rapport au clavier AZERTY respectivement pour le clavier GAG (résultats calculés de manière prédictive, qui donnent un gain de vitesse de l'ordre de 50% par rapport au clavier AZERTY) et le système KeyGlass (résultats obtenus après expérimentation avec des sujets handicapés qui montrent une diminution d'environ 25% des distances parcourues pour saisir du texte).

Mots-clés: Clavier logiciel, handicap moteur, optimisation, évaluation