THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

INFORMATIQUE - Interaction Homme-Machine

Présentée et soutenue par :
Bruno MERLIN

le : mardi 21 juin 2011

Titre :

Méthodologie et instrumentalisation pour la conception et l'évaluation des claviers logiciels

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
IRIT - UMR 5505

Directeur(s) de Thèse :

Pr. Philippe PALANQUE, M. Mathieu RAYNAL

Rapporteurs :

Pr. Franck POIRIER, M. Benoît MARTIN

Autre(s) membre(s) du jury

Pr. Patrick GIRARD, Mme Nadine VIGOUROUX

Remerciements

Je remercie en premier lieu Philippe Palanque et Mathieu Raynal, mes directeur et encadrant de thèse, pour m'avoir accompagné à la fois scientifiquement et administrativement dans la réalisation de ces travaux. Je leur fais gré par ailleurs pour m'avoir accordé leur confiance malgré la complexité du contexte international dans laquelle celle-ci était amenée à se dérouler.

Je souhaite également remercier vivement les membres du jury, Messieurs Frank Poirier et Benoît Martin en tant que rapporteurs et Nadine Vigouroux et Patrick Girard en tant qu'examinateurs, pour m'avoir, en premier lieu, fait l'honneur d'accepter de participer à l'évaluation de mes travaux, et, en second lieu, pour avoir, par leurs travaux respectifs, inspirés une partie de cette thèse. Je remercie par ailleurs Nadine Vigouroux ainsi que Philippe Truillet pour avoir été, quelques années plus tôt, à l'origine de ma vocation et de ma formation scientifique.

Je souhaite aussi remercier tout particulièrement : Raïlane Benhacène qui par son amitié et sa confiance m'a permis de prendre la pleine mesure de mes capacités, stimulant de ma part une attitude volontaire et persévérante ; et Cleidson qui par son soutien, ses conseils et sa patience envers mon portugais initialement précaire, a facilité grandement mon intégration dans le monde académique brésilien.

Je remercie également tous mes collègues français et brésiliens qui, au-delà de m'avoir apporté leur soutien moral et parfois technique, se sont aussi fréquemment adonnés au rôle de cobaye des nombreuses expérimentations et pré-expérimentations menées dans le cadre de cette thèse... Et encore pardon aux quatre valeureux qui ont dûs affronter les 80 sessions nécessaires à l'évaluation du clavier Multi-Layer!

Je souhaite aussi remercier tous mes amis qui, au-delà de m’avoir apporté la joie de leur présence et pardonné mes absences au cours des périodes les plus exigeantes de mes travaux, on toujours été présents pour partager les bons moments comme les moments les plus difficiles.

Je remercie enfin les membres de ma famille, mes parents et frères avant tout, pour tout ce qu’ils ont pu m’apporter bien avant le début de cette thèse et bien avant même que j’en ai conscience.

Je dédie ces dernières lignes à Lise avec qui je partage mes jours depuis de nombreuses années déjà, et qui contribue très largement à rendre ces jours *felizes*.

Table des matières

Tables des figures

Liste des tableaux

Liste des exemples

Introduction

L'utilisation des claviers logiciels s'est massivement répandue au cours de ces dernières années. Ils étaient il y a peu de temps encore dédiés presque uniquement à des fins d'accessibilité aux handicapés. Mais l'apparition des PDA puis des téléphones mobiles à écran tactile a considérablement contribué à populariser et généraliser leur usage. Cependant, comme nous pouvons l'observer quotidiennement, les alternatives existantes au traditionnel clavier AZERTY (hormis le clavier téléphonique) restent très marginalement utilisées. Bien que celui-ci soit théoriquement et expérimentalement [Goldberg 93, Zhai 00] reconnu comme largement sous optimal, le clavier AZERTY reste le moyen de saisie, sinon préféré, au moins spontané de la très grande majorité des utilisateurs.

Comme nous l'avons implicitement évoqué, cette problématique n'est néanmoins pas nouvelle. De nombreux projets se sont attachés à proposer des alternatives évaluées théoriquement [Raynal 05b] ou mesurées expérimentalement [MacKenzie 99] comme très avantageuses. Pourtant, malgré des perspectives de multiplier parfois par deux la vitesse de saisie [Textware 98], elles n'ont trouvé qu'un écho très limité auprès du grand public.

Chaque jour plus présente autour de nous, l'informatique mobile compte donc toujours parmi les nombreux défis qu'elle doit relever en termes d'interaction homme-machine, l'amélioration de l'efficacité de la saisie de texte.

Au cours de ce manuscrit nous allons, dans une première partie, nous interroger sur les raisons qui freinent l'avènement de solutions alternatives. Sans nier qu'il existe une part culturelle liée à l'usage du clavier AZERTY (versus QWERTY dans les milieux anglophones, QWERTZ, etc.), nous nous questionnerons sur les métriques habituellement utilisées pour évaluer et comparer les différents outils et questionnerons la notion de **performances** obtenues avec un clavier logiciel.

Pour expliquer le questionnement autour de l'évaluation des performances, nous partons du parallèle suivant :

- En termes de moyen de déplacement, les Formules 1 sont des instruments très **performants** dans la mesure où elles permettent d'atteindre des vitesses sur piste extrêmement élevées. Néanmoins, en raison de leur complexité, seuls quelques individus très entraînés sont en mesure de les piloter. Par ailleurs, leur fragilité les rendrait totalement inadaptées à un usage quotidien sur une route quelconque.

- Bien que beaucoup moins rapide, une voiture routière est un instrument très **performant** dans la mesure où elle permet à la très grande majorité des individus de

se déplacer avec une vitesse moyenne et une relative sécurité. En outre, un utilisateur accoutumé à un véhicule s'adapte aisément au modèle concurrent.

On s'accordera à dire que comparer une voiture routière et une Formule 1 n'aurait en soit pas beaucoup de sens dans la mesure où, bien qu'étant deux véhicules à quatre roues partageant certaines caractéristiques communes, ils sont conçus avec deux finalités radicalement différentes. Les critères de comparaison qui visent à comparer les Formules 1 et les critères visant à comparer les voitures sont différents et conduisent à produire des artefacts différents.

Au regard de cet exemple, notre interrogation est de savoir si la mesure de la compétitivité d'un clavier logiciel, son évaluation, ne nous conduit pas à ne créer que des « Claviers Formule 1» : des claviers efficaces pour un utilisateur entraîné mais sans perspective d'être appréhendés par une population plus globale parce que le pas à franchir pour maîtriser complètement l'outil freine considérablement sa prise en main et son acceptation (cf. Figure 1).

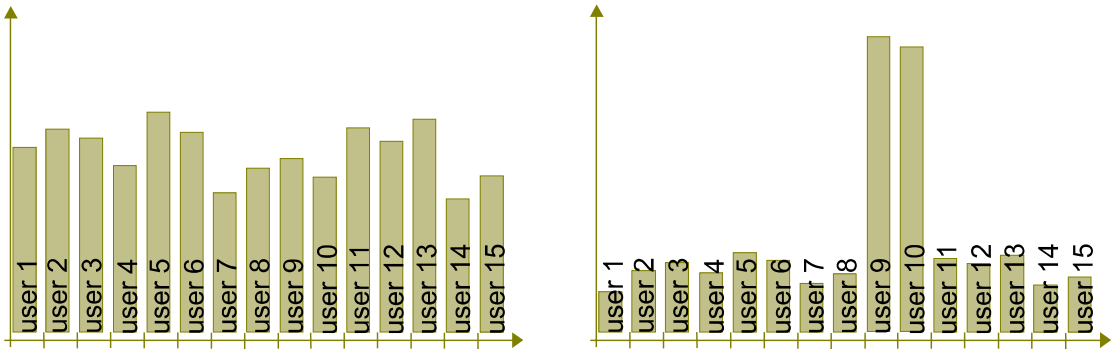


Figure 1. Quel est le plus performant des deux artefacts ?

Ainsi, nous observerons que les démarches traditionnellement utilisées pour comparer un large ensemble de claviers [MacKenzie 99, Zhai 02a] privilégient une unique métrique : la vitesse de saisie. Ces démarches ne fournissent de perspectives ni sur l'utilisabilité d'un outil au quotidien ni sur l'acceptabilité d'un nouvel outil par une population.

Sur la base de travaux préliminaires, et après avoir expliqué les limitations actuelles des démarches théoriques et expérimentales, nous justifierons de l'importance de prendre en compte, dans les évaluations et comparaisons, les premiers usages au moyen des nouveaux claviers. Nous tenterons d'améliorer le calcul des modèles prédictifs existants et conclurons sur la difficulté de définir des modèles prédictifs génériques et pertinents comparant les premiers usages.

Nous justifierons en conséquence de l'importance de la démarche expérimentale et aborderons la nécessité de fournir de nouveaux indicateurs heuristiques destinés à mieux anticiper et comparer les perspectives d'acceptabilité par une population.

De manière à concrétiser cet objectif de faciliter les démarches expérimentales et plus généralement d'accompagner l'évaluation des claviers, la seconde partie de nos travaux s'est orientée vers l'amélioration de la plate-forme d'expérimentation E-Assiste [Raynal 07c].

L'objectif initial de cette plate-forme était de fournir un ensemble de briques logicielles : panneaux de consignes ; bandeaux de présentation des informations à saisir durant l'expérimentation ; outils d'acquisition des données ; protocole de communication entre le clavier, le bandeau de présentation et les outils d'acquisition ; et enfin, quelques claviers développés pour différentes expérimentations réalisées à partir de la plate-forme. Ces briques logicielles permettent de monter une expérimentation et d'encadrer le protocole d'expérimentation (notamment organiser l'ordonnancement des exercices et des consignes). Néanmoins, l'assemblage de ces briques logicielles requiert une étape de programmation nécessitant l'intervention d'un développeur familier avec l'environnement pour accompagner la mise en œuvre d'une expérimentation.

Notre premier axe de travail a été la modélisation (sous format XML) des protocoles d'expérimentations, afin non seulement de faciliter l'intégration des briques logicielles et le montage des expérimentations, mais également de formaliser la gestion des utilisateurs, groupes d'utilisateurs et le contre-balancement des exercices sur plusieurs sessions d'expérimentation.

Par ailleurs, les claviers logiciels ne sont plus de simples jeux de boutons juxtaposés imitant le clavier physique. Désormais de nombreux claviers logiciels profitent d'un système de prédiction pour optimiser dynamiquement la saisie (en proposant par exemple des listes de complétion [Darragh 90, Masui 99]), d'autres utilisent des techniques d'interaction alternatives, telles que le geste, pour enrichir la sémantique de l'interaction avec la touche (par exemple les claviers à touches ambiguës dont la saisie est discriminée par reconnaissance de geste [Nesbat 03, TouchPal[1], Merlin 10b]), d'autres encore font abstraction d'une interaction avec des touches et proposent une interaction globale avec le clavier sous forme, par exemple, de saisie continue [Ward 00, Zhai 02a]. Concevoir et développer un clavier à l'usage d'une expérimentation est donc une tâche de plus en plus complexe qui inclut la connaissance des modèles linguistiques et le design de l'interaction.

Notre second axe d'amélioration de la plate-forme c'est ainsi logiquement orienté vers comment faciliter le design de ces claviers logiciels. A travers l'analyse d'un grand nombre de claviers, nous avons en premier lieu défini et organisé autour d'une architecture commune les identités fonctionnelles qui composent un clavier logiciel. Sur la base de cette architecture, nous proposons un langage XML (KeySpec) de spécification de claviers et un kit de développement permettant de compiler et exécuter les claviers logiciels. Nous avons finalement permis l'utilisation par E-Assist des claviers spécifiés par KeySpec.

Par ailleurs, malgré la relativité des résultats fournis par les modèles prédictifs (relativité identifiée en première partie) nous montrerons également comment les éléments relatifs à ces modèles peuvent être intégrés dans le langage de spécification des claviers de manière à obtenir rapidement des données théoriques sur ceux-ci.

Enfin, nous montrerons comment nous avons intégré des outils d'analyse des données de l'évaluation expérimentale et heuristique de manière à fournir une plate-forme d'évaluation complète des claviers logiciels.

A travers la réflexion menée au cours de la première partie, complétée par les simulations effectuées grâce au support de la plate-forme, nous montrerons que, pour envisager une

[1] *www.cootek.com*

amélioration des performances à long terme avec un nouveau clavier, il est indispensable d'obtenir de bonnes performances à court terme. Orientés par cette observation, nous proposerons des stratégies pour l'amélioration des claviers logiciels. Ces trois stratégies visent un but commun : améliorer la saisie tout en facilitant l'appropriation du nouveau clavier en conservant de prime abord l'apparence d'un clavier standard.

Le premier clavier, SpreadKey, élaboré sur la base du clavier AZERTY, utilise un système de prédiction pour recycler les touches dont le caractère a une très faible probabilité d'être saisi en fonction des précédents caractères saisis. Les caractères recyclés sont remplacés par des caractères dont la probabilité d'être saisis est élevée. L'algorithme de recyclage vise, tout d'abord, à créer des groupes de touches contenant le même caractère de manière à faciliter le pointage en proposant des touches plus larges pour les caractères les plus probables. Il vise dans un deuxième temps à améliorer la saisie en réduisant la distance parcourue par le dispositif de pointage. Il tend à proposer ces mêmes caractères les plus fréquents sur des touches situées entre la dernière touche frappée et la touche contenant naturellement le caractère (cf. Figure 2).

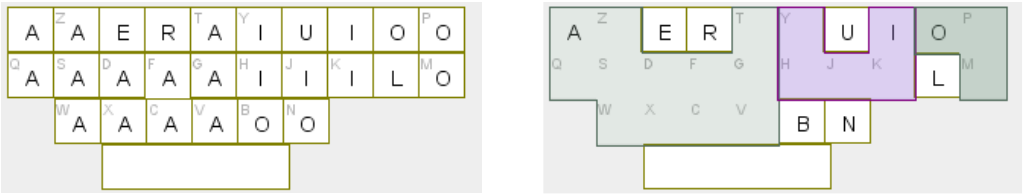


Figure 2. SpreadKey I et II

Cependant, et malgré des résultats positifs par rapport aux utilisateurs handicapés moteurs, SpreadKey [Merlin 09a, Merlin 10a] ne s'est pas avéré très efficace dans le contexte d'un usage général. Les nombreux changements dynamiques affectent en effet la perception de l'utilisateur et ralentissent l'enchaînement de touches. En conséquence, l'usage du clavier est également fatiguant.

Pour résoudre ce problème, la seconde stratégie vise à modifier l'espace moteur sans modifier l'espace visuel. L'apparence du clavier reste la même, en revanche l'accès aux touches est facilité par pointage sémantique en fonction de la probabilité de saisie des caractères. La vitesse du pointeur est modifiée en fonction de la direction du pointeur et de la probabilité des caractères de manière à faciliter l'accès aux caractères les plus probables.

La dernière stratégie vise à accompagner progressivement l'utilisateur vers l'utilisation d'une nouvelle distribution de touches. Plusieurs projets de recherche ont proposé des distributions [Zhai 00, Raynal 05b] a priori mieux adaptées que l'agencement AZERTY (versus QWERTY). Ces distributions sont optimisées en fonction des caractéristiques linguistiques et pour l'usage du clavier via un pointeur unique (souris ou stylet par exemple). Néanmoins ces nouvelles distributions, bien que permettant des performances à moyen / long terme très intéressantes, sont difficilement acceptées par un utilisateur qui, dans un premier temps, perd ces repères par rapport au clavier AZERTY (et en conséquence perd en efficacité).

L'idée du clavier est de bénéficier d'un nécessaire rétro contrôle visuel dans l'usage d'un clavier logiciel. Dans le cadre de l'usage d'un clavier physique, l'accès aux touches est mécanique et la permutation de deux touches à des conséquences importantes sur la saisie. Dans le cadre de l'usage d'un clavier logiciel, l'utilisateur doit nécessairement corriger la trajectoire du pointeur en fonction d'une recherche visuelle de la touche ciblée. Notre hypothèse est que dans ce contexte, la permutation de deux touches à un impact quasi négligeable sur les performances. L'objectif est ainsi de procéder, sur une longue période, à

des permutations successives entre les caractères pour amener progressivement l'utilisateur d'un clavier AZERTY vers une autre distribution de touche sans perte d'efficacité.

Enfin, nous observerons que, si de nombreux travaux se concentrent exclusivement sur l'optimisation de la saisie en ne considérant que les 26 caractères de l'alphabet latin plus l'espace, des gains simples et importants sont envisageables par l'optimisation de la saisie des autres caractères tels que majuscules, caractères accentués, etc.

Première Partie

Méthode d'évaluation des claviers logiciels

De précédents travaux [Raynal 07c] nous ont permis d'illustrer que les outils d'évaluation, habituellement utilisés pour effectuer une comparaison des vitesses de saisies théoriques des claviers, s'avéraient désormais insuffisants. Les modèles comme le *upper-bound* [Soukoreff 95] ou KLM [Card 80, Card 83] appliquant les lois prédictives mécaniques (loi de Fitts [Fitts 54, Mackenzie 92b]) et de sélection d'une cible (loi de Hick-Hyman [Hick 52, Hyman 53]) ne permettent pas de rendre compte de l'utilisation d'outils plus sophistiqués que nous nommerons d'une manière générale **claviers complexes**. L'utilisation d'une liste de complétion [Masui 99, Tanaka-Ishii 02], d'autres touches dynamiques du clavier [Merlin 09b, Raynal 07c], ou encore l'utilisation de claviers ambigus [Levine 87, Lesher 98] ne relèvent plus du simple pointage mais engagent d'autres processus cognitifs plus élaborés, que nous ne savons pas modéliser de façon pertinente, et qu'il serait pourtant nécessaire d'intégrer dans les modèles pour avoir une idée plus représentative des performances. Face à la complexité des processus engagés, nous questionnerons donc en premier lieu les limites de ces évaluations théoriques.

Par ailleurs, nous questionnerons la finalité même des méthodologies d'évaluations proposées, quelles soient expérimentales ou théoriques. En effet, dans la perspective d'amener une population d'utilisateurs à améliorer son efficacité en matière de saisie de texte, nous nous interrogerons sur le sens de ne considérer essentiellement que les performances maximales accessibles à long terme. Nous discuterons de la nécessité d'établir, *a minima*, des critères complémentaires permettant de qualifier la prise en main, les coûts physiques et cognitifs, l'apprentissage, etc., relatifs à l'usage du clavier, et ceci pour différentes classes d'utilisateurs, dans différents contextes d'utilisation, que ce soit pour des utilisateurs intensifs ou occasionnels de l'outil.

Dans un premier chapitre, nous présenterons les méthodes actuelles, théoriques et expérimentales, d'évaluation des systèmes de saisie de texte consistant essentiellement à comparer les claviers sur la base d'une donnée principale : les performances d'un utilisateur expert (ou éventuellement d'un novice). Dans les deux chapitres suivants, sur la base de travaux préliminaires, nous expliquerons les limites de ces paradigmes d'évaluations et justifierons en premier lieu le manque d'objectivité de la notion d'expert face aux nouveaux paradigmes de saisie de texte (Chap. 2), et, en second lieu, la relativité de la notion d'utilisateur novice (Chap. 3). Nous discuterons enfin de l'adéquation nécessaire entre les critères et la finalité des évaluations et proposerons de nouveaux critères d'évaluation (Chap. 4).

Chapitre 1 - Évaluation des systèmes de saisie de texte : les métriques mesurées à travers les évaluations théoriques et expérimentales

Les méthodologies scientifiques habituellement utilisées pour évaluer les claviers logiciels sont de deux natures : méthodologie expérimentale; ou modélisation mathématique et calcul théorique des performances en fonction de lois prédictives modélisant le comportement humain d'un point de vue moteur et cognitif[2].

Une démarche d'évaluation type est la réalisation d'une étape de modélisation calculant les performances maximales accessible avec un clavier, le *upper-bound* [Soukoreff 95] (établi comme les performances théoriques d'un expert), suivie d'une validation des performances calculées par expérimentation [Costagliola 09, Merlin 09a, Merlin 10a] dans le cas où la première étape donne satisfaction. L'ensemble de ces démarches convergent avec pour objectif la mesure des performances d'un utilisateur expert.

Nous présenterons en premier lieu les lois prédictives génériques (I) et leur application dans le cadre de l'évaluation des claviers logiciels (II). Puis nous discuterons des méthodologies d'évaluation expérimentales (III) et conclurons sur les objectifs convergeant de ces méthodologies (IV).

Section I - Lois prédictives

Plus qu'un moyen d'évaluation des claviers logiciels, l'utilisation des lois prédictives, telles que les lois de Fitts [Fitts 54] (adaptée pour l'IHM par MacKenzie [MacKenzie 92a]) et Hick-Hyman [Hick 52, Hyman 53] présentées ici, visait initiallement à évaluer le temps d'exécution de tâches physiques, puis à évaluer de manière quantitative les IHM ou au moins le temps d'exécution de certaines tâches au sein d'une IHM. Adaptées pour l'évaluation de tâches répétées, ces lois ne permettent pas, en revanche, de traiter efficacement l'exécution de tâches engageant des processus cognitifs complexes et d'autres aspects ergonomiques des IHM abordés, par exemple, par les critères de Ravden et Johnson [Ravden 89], de Bastien et Scapin [Bastien 93], ou de Nielsen [Nielsen 90, Nielsen 94].

I.1. Prédiction du temps de pointage d'une cible (loi de Fitts)

Au cours de ces travaux en psychologie expérimentale, Robert S. Woodworth est le premier à identifier, en 1899, l'incompatibilité entre une grande précision et une grande rapidité d'exécution d'un mouvement dans l'action de pointage d'une cible [Woodworth 99]. Ces travaux, ainsi que les travaux de Shannon [Shannon 48] sur la théorie de l'information, ouvrent la voie aux travaux de Fitts [Fitts 54] visant à quantifier la difficulté d'une tâche de pointage.

En analogie avec la relation de Shannon, celle-ci décrivant la capacité C de transmission d'un message dans un environnement bruité, Fitts modélise la tâche de pointage comme la transmission d'un message dans un système perceptuo-moteur. Cette transmission est bruitée par la précision du geste à réaliser et la difficulté de la tâche.

[2] modélisant essentiellement le temps de recherche d'une touche et le temps de pointage

Shannon propose la relation suivante : pour un signal D et considérant un bruit W, la capacité C de transmission d'un message s'exprime par la relation :

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

Fitts transpose cette relation et propose un indice de difficulté (ID) qualifiant la difficulté de réaliser un mouvement de distance D dont le but est d'atteindre une cible de largeur W. L'indice de difficulté est matérialisé par la relation suivante :

$$\boldsymbol{ID} = \log_2(\frac{2*\boldsymbol{D}}{\boldsymbol{W}})$$

Fitts relie cet indice de difficulté et le temps d'exécution de la tâche de pointage à travers trois expériences consistant à effectuer un mouvement de va et vient entre deux cibles (cf. Figure 3). La loi résultant, qui porte son nom, prédit le temps d'exécution T de la tâche de pointage à travers la formule suivante :

$$\boldsymbol{T} = \boldsymbol{a}+\boldsymbol{b}*\boldsymbol{ID} = \boldsymbol{a}+\boldsymbol{b}*\log_2(\frac{2*\boldsymbol{D}}{\boldsymbol{W}})$$

où

- T est exprimé en seconde
- D représente l'amplitude du mouvement à effectuer
- W la taille de la cible à atteindre
- a et b sont des constantes obtenues par régression linéaire. Ces constantes dépendent des propriétés kinesthésiques du dispositif de pointage et des capacités psychomotrices de la population concernée par l'expérimentation.

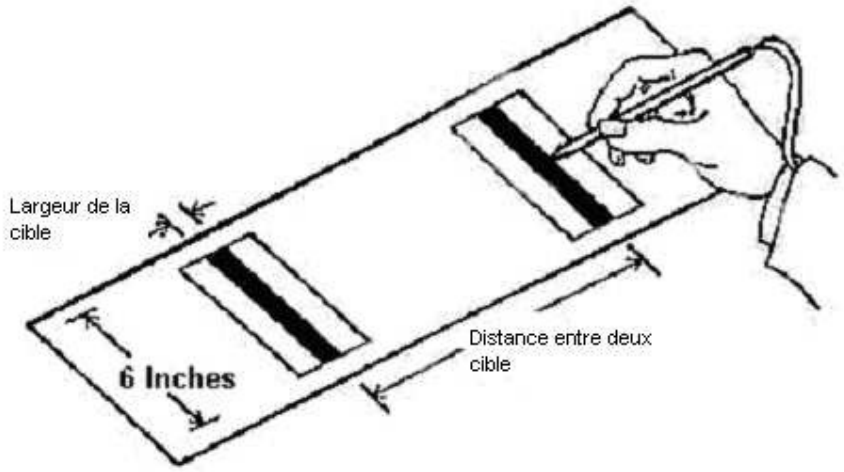


Figure 3. Expérience conduite par Fitts

Dans un second temps, Fitts et Peterson [Fitts 64] démontrèrent, par une nouvelle expérience, que la loi ne s'appliquait pas uniquement à une répétition de mouvements, mais également à une tâche discrète de pointage d'une cible.

Notons que le rapport D/W rend le temps d'exécution du pointage insensible à l'échelle du pointage. Cette indépendance est contestable et contestée sur la base de l'étude de l'évolution de la vitesse d'exécution du pointage [Guiard 09], notamment lorsque nous abordons les limites de distance petites ou grandes.

I.2. La loi de Fitts en IHM

A la fin des années 70, Card & al. [Card 78] étendirent les résultats précédents en montrant que la loi restait valide dans le cadre d'un pointage indirect dans un contexte virtualisé. Le pointage d'une cible à l'écran d'ordinateur, via un dispositif de pointage comme la souris, répond ainsi aux mêmes lois que le pointage par manipulation directe d'un artefact physique en direction d'une cible physique. Ces résultats furent par la suite confirmés par MacKenzie [MacKenzie 92b] puis Douglas [Douglas 97]

Une reformulation de la loi de Fitts, proposée par MacKenzie [MacKenzie 95] et plus proche de la relation de Shannon, est plus couramment utilisée en IHM car elle est considérée comme plus robuste à différentes situations. En IHM, le temps de pointage est donc fréquemment établi par la formule suivante :

$$\boldsymbol{T = a + b * ID = a + b * \log_2(\frac{D}{W} + 1)}$$

où a, b, W et D ont une fonction identique à leur rôle dans la formulation faite par Fitts.

Néanmoins, la cible engagée dans la tâche de pointage proposée par Fitts a des caractéristiques particulières : il s'agit d'une bande dont seule la largeur est considérée, et où la trajectoire du pointeur est perpendiculaire à la longueur de cette cible (colinéaire à la largeur). La tâche de pointage est donc effectuée en une dimension. Cette caractéristique se retrouve finalement assez peu en IHM où le pointage est effectué sur des cibles en deux dimensions : les cibles sont souvent des boîtes rectangulaires (voire des formes plus complexes) et les trajectoires aléatoires, dépendant de la tâche antérieure et ainsi de la position courante du curseur.

Bien que dans plusieurs travaux [Card 78, Jagacinski 85], l'étude se satisfasse de l'approximation consistant à considérer le pointage d'une cible en deux dimensions par la simple application de la loi de Fitts destinée à un pointage en une dimension, plusieurs autres études ont proposé des stratégies pour améliorer cette approximation [Gillan 90, MacKenzie 91a, MacKenzie 92a, Accot 03, Appert 08b, Yang 10]. Ces stratégies consistent, dans le cas de cibles rectangulaires à considérer comme bruit (W) une relation entre les deux dimensions de la cible comme, par exemple, le minimum entre la largeur et la hauteur, la somme des deux, ou encore l'aire du rectangle. Une autre stratégie, notamment appliquée pour des cibles plus complexes, consiste à tenir compte de l'angle de pénétration du pointeur dans la cible. Elle considère que l'utilisateur, à partir d'un point de départ initial connu, va viser le centre de la cible et propose comme valeur W l'intersection entre la trajectoire du pointeur et la cible (cf. Figure 4). Cette valeur est donc différente pour chaque pointage.

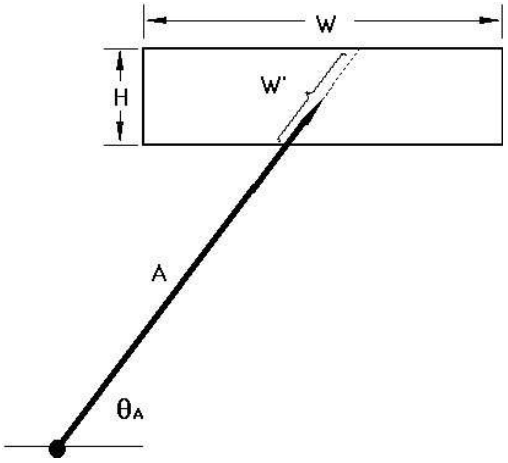


Figure 4. Modélisation de la taille de la cible

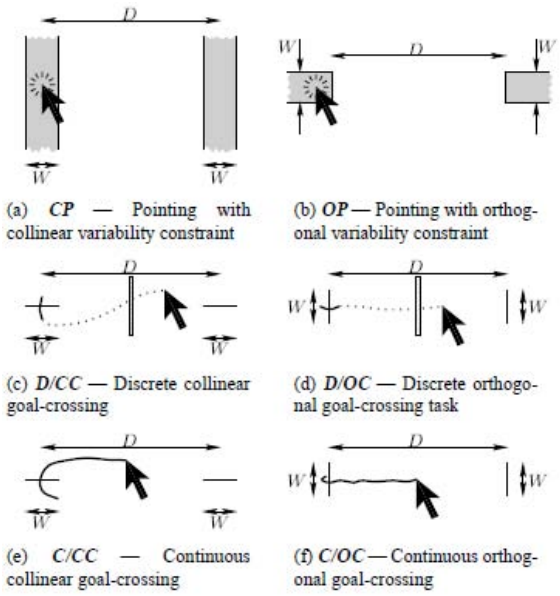


Figure 5. Modélisation des tâches de franchissement

Par ailleurs, étendant les recherches entre la nature de la cible et la modélisation de la tâche, Accot et Zhai [Accot 02] proposent une taxonomie et des modèles pour les différentes tâches de franchissement contraint d'une frontière (cf. Figure 5).

I.3. Prédiction du temps de recherche visuelle d'une cible

S'appuyant, de la même manière que Fitts, sur les bases de la théorie de l'information proposée par Shannon, Hick en 1952 [Hick 52] et Hyman en 1953 [Hyman 53], indépendamment l'un de l'autre, proposent une relation exploitée[3] pour déterminer le temps de recherche visuelle d'une cible parmi N cibles affichées.

La loi de Hick-Hyman explicite plus largement le temps nécessaire pour prendre une décision entre N possibilités équiprobables. Elle suggère un temps de réponse logarithmique en fonction du nombre de possibilités offertes à l'utilisateur. Cette réponse logarithmique, et donc non linéaire, est justifiée considérant que l'utilisateur ne va pas évaluer les choix linéairement un par un, mais catégoriser les choix de façon dichotomique par familles de choix, puis par sous-familles, sous-sous-familles, etc., parcourant ainsi un arbre binaire de recherche jusqu'à atteindre une feuille de l'arbre en éliminant la moitié des possibilités à chaque nœud de l'arbre.

Card, Moran et Newell [Card 83], suggèrent que le temps dépend du nombre de choix N + 1 et non uniquement de N, justifiant que le +1 correspond à la prise de décision de répondre ou non. Cette variante a également le mérite de corriger une anomalie mathématique donnant un temps de décision infiniment négatif (au lieu de 0) lorsqu'il n'y a qu'une alternative et qu'aucune décision n'est donc à prendre (N = 0).

Certains modèles considèrent que la recherche visuelle d'une cible peut donc être assimilée à cette activité de prise de décision entre N possibilités équiprobables, par exemple lorsque l'utilisateur doit cibler un parmi N items graphiquement voisins et répartis uniformément. Ces

[3]Nous observerons plus tard que cette utilisation est contestée par Sears [Sears 01] puis Pavlovych [Pavlovych 04] et Das [Das 08], très certainement à bon escient

modèle établissent, en conséquence, le temps de recherche TR par la loi de Hick-Hyman (corrigée par Card et al.) :

$$\boldsymbol{TR} = \boldsymbol{b} * \log_2(\boldsymbol{N}+1)$$

Où :

- N est le nombre de choix
- b est déterminé par régression linéaire et est inhérent à une population et à la nature du choix. Néanmoins, dans la littérature, il est généralement attribué un coéficient de 1/4,9 lorsque l'utilisateur est novice dans une tâche et un coéfficient de 1/7 lorsque l'utilisateur est expert.

Cependant, comme nous l'expliquerons dans le troisième chapitre, cette utilisation est largement controversée [Sears 01, Das 08] et nous nous en dissocions.

Landauer et Nachbar [Landauer 85] montrèrent, en revanche, que les lois de Hick-Hyman pouvaient s'appliquer à déterminer le temps de sélection d'un item dans un menu organisé alphabétiquement dont le contenu est maîtrisé par l'utilisateur.

I.4. KLM et GOMS

Les modèles GOMS [Card 83, Gray 93, John 96] et KLM [Card 80] visent à découper une tâche en un ensemble de sous-tâches élémentaires pouvant être de natures cognitives (par exemple réflexion concernant le processus à mettre en œuvre pour exécuter la tâche ou la sous-tâche, attente d'un feedback, réponse à un stimulus) ou effectives (déplacement du curseur, pression du bouton de la souris, d'une touche du clavier, etc.). Le temps de chaque sous-tâche est évalué empiriquement (cf. Tableau 1) et la somme des temps de chaque sous-tâche détermine le temps d'exécution global de la tâche.

Pression d'une touché ou d'un bouton		
	Utilisateur expert	0.08 seconde
	Bon utilisateur	0.12 seconde
	Utilisateur moyen	0.20 seconde
	Saisie discrète d'une touché	0.50 seconde
	Utilisateur débutant	1.20 seconde
Pointage avec la souris (sans le clic)		1.10 seconde
Mouvement de la main en direction de la souris (ou l'inverse)		0.40 seconde
Préparation mentale		1.35 seconde

Tableau 1. Evaluation empirique des sous-tâches [Card 80] (traduit de l'anglais)

La première critique faite à l'égard du modèle est que la tâche de pointage, matérialisée par un coût fixe, est indépendante de sa complexité. Le modèle ne permet en conséquence pas de différencier le pointage d'une cible distante, plus petite, plus large, etc. Il est néanmoins possible d'adapter le modèle en calculant le temps de pointage par l'intermédiaire de la loi de Fitts [Dunlop 00].

Section II - Evaluation théorique des performances d'un clavier

Les claviers logiciels, sollicitant une tâche simple et répétée de recherche d'un item suivi de son pointage, sont devenus un terrain d'application idéal pour étudier et comparer les lois prédictives ainsi que leurs paramètres. Réciproquement, l'évaluation des claviers logiciels a hérité de modèles spécifiquement adaptés, facilitant la comparaison théorique des différents claviers logiciels (et, essentiellement et initialement de différentes distributions de touches).

II.1. Modèle de Soukoreff et MacKenzie

La démarche initiale de Soukoreff et MacKenzie [Soukoreff 95] vise à fournir un outil de comparaison statique des différents claviers. Il considère deux grandeurs opposées : les performances théoriques d'un expert (*upper-bound*) et celles d'un novice (*lower-bound*). Ces valeurs sont obtenues en décomposant la tâche de saisie en deux activités : une activité de pointage et une activité de recherche d'une cible. Le temps nécessaire à cette seconde est relatif au niveau d'expertise de l'utilisateur par rapport à l'usage du clavier.

Dans le modèle proposé, le temps de ces deux activités est déterminé par l'intermédiaire des lois de Fitts en ce qui concerne le pointage et l'utilisation de la loi de Hick-Hyman en ce qui concerne le temps de recherche visuelle des caractères.

Pour caractériser le temps de saisie, Soukoreff et MacKenzie proposent d'évaluer le temps nécessaire à la saisie de chaque caractère, au moyen de la loi de Fitts, en tenant compte des différentes possibilités pour la position initiale du pointeur. Cette position initiale est relative au précédent caractère saisi. Ainsi, le temps moyen de saisie d'un caractère est relatif à toutes les combinaisons possibles de deux caractères (bigrammes) pondérées par leur fréquence d'apparition dans la langue considérée.

Le temps minimum MT est donc exprimé par la loi :

$$\overline{MT} = \sum_{i \in \Psi} \sum_{j \in \Psi} P_{ij} T_{ij}$$

Où, en considérant Ψ l'alphabet contenant l'ensemble des caractères présents sur le clavier :

- Pij est la fréquence d'apparition du bigramme $C_I C_J$ dans la langue considérée, où $(C_I, C_J) \in \Psi^2$;
- Tij est le temps de saisie du caractère C_J en considérant que le précédent caractère saisi est I.

Le temps T_{ij} est lui-même évalué par la loi de Fitts :

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

Où :

- D_{ij} est la distance séparant la touche contenant le caractère C_I et la touche contenant le caractère C_J ;
- W la largeur de la touche ciblée contenant C_J.

Notons que Soukoreff et MacKenzie considère un temps particulier, déterminé expérimentalement, pour la répétition d'un même caractère (Tjj = 0,153s).

Pour un expert, le modèle établit que le temps de recherche d'une cible est nul considérant que l'enchaînement des frappes est spontané. La saisie se résume donc à une succession de pointages et le temps de saisie moyen (MT) est donc retenu comme valeur caractéristique des performances d'un expert.

En revanche, pour un novice, le modèle considère que l'utilisateur ignore totalement la répartition des touches du clavier, toutes les touches sont considérées comme équiprobables. Le temps de saisie est donc grevé par une valeur constante, RT, que Soukoreff et MacKenzie établissent au moyen de la loi de Hick-Hyman, interprétée comme correspondant au temps de recherche d'une touche parmi les N touches composant le clavier.

Notons que dans l'application du modèle, N est souvent limité à 27 (les 26 caractères de l'alphabet latin plus l'espace) de manière à faciliter les comparaisons expérimentales et internationales (les anglo-saxon ne disposant notamment pas de caractères accentués). Le modèle de Soukoreff et MacKenzie considère donc généralement le temps de recherche RT tel que :

$$\boldsymbol{RT} = \boldsymbol{b} * \log_2(\boldsymbol{N}+1) = 0{,}951\text{s}.$$

MT et MT + RT bornent les performances entre le cas le plus défavorable, où, pour chaque caractère, l'utilisateur doit rechercher la touche sans aucune heuristique pour la localiser, et le cas le plus favorable où la vitesse est uniquement limitée par les capacités physiologiques d'exécution des mouvements.

Ces deux bornes, le *lower-bound* (borne inférieure) et le *upper-bound* (borne supérieure) sont généralement exprimées en mots par minute (WPM – word per minute). Étant établi que la longueur moyenne d'un mot est de 5 caractères [Gentner 83], le lower-bound ($WPM_{min)}$ et le upper-bound ($WPM_{max)}$ sont exprimés par les calculs suivants :

$$\boldsymbol{WPM}_{\min} = \frac{60}{5 \times (\overline{\boldsymbol{MT}} + \boldsymbol{RT})} \qquad \text{et} \qquad \boldsymbol{WPM}_{\max} = \frac{60}{5 \times \overline{\boldsymbol{MT}}}$$

La seule véritable variable du modèle est MT caractérisant finalement l'efficacité d'une disposition de touches (ceci par rapport à une langue, une population cible, un dispositif de pointage et une géométrie de touches). Le modèle est donc particulièrement efficient, et exploité, pour comparer des claviers tels que Metropolis [Zhai 00], Fitaly [Textware 98, Zhai 02a], GAG [Raynal 05b], etc., différents des claviers AZERTY/QWERTY par la distribution des caractères. En revanche, dans la mesure où le seul processus cognitif considéré est le temps de recherche d'une cible, le modèle ne peut être utilisé dans l'évaluation de claviers plus complexes, et notamment ceux faisant intervenir des aspects dynamiques (listes de complétion ou KeyGlass [Raynal 05a] par exemple).

Par ailleurs, et comme nous l'étudierons de manière plus détaillée ultérieurement (cf. Chapitre 3 -), le cas le plus défavorable (lower-bound) est pessimiste [MacKenzie 95, MacKenzie 01b, Pavlovych 04] par rapport aux valeurs expérimentales observées. En outre, d'autres limites du modèle seront évoquées ultérieurement.

II.2. KLM et évaluation des claviers logiciels

Plusieurs travaux ont proposé d'adapter le modèle KLM à l'évaluation des claviers logiciels. Ces travaux visaient à évaluer les performances dans des contextes tels que l'usage de claviers virtuels par des utilisateurs handicapés [Koester 94], ou l'usage de claviers téléphoniques physiques [Dunlop 00] ou virtuels [Holleis 07].

L'avantage du modèle KLM est qu'il offre plus de souplesse théorique que le modèle de Soukoreff et MacKenzie pour décrire la tâche et intégrer des processus interactifs plus complexes que la simple succession de pointages. Il permettrait par exemple d'intégrer une charge cognitive relative à la consultation d'une liste de complétion. La problématique centrale de l'application du modèle repose sur la décomposition de la tâche en sous tâches et sur l'évaluation des temps à affecter pour chaque sous-tâche[4].

La décomposition d'une tâche en sous-tâches peut être critiquable, mais l'obtention du temps d'exécution pour chaque sous-tâche relève également d'un problème particulièrement complexe. En effet, sauf dans certains cas précis pour lesquels le temps d'exécution d'une sous-tâche a été mesuré, les valeurs dépendent globalement du dispositif utilisé, de l'acuité physique et mentale de la population cible et du contexte général d'utilisation (mobilité par exemple). La combinatoire est importante. Par ailleurs, l'activité mentale liée notamment à la recherche d'une touche diffère forcément entre un utilisateur novice et un utilisateur expert.

Bien entendu, ces différentes valeurs peuvent être approximées, mais pour disposer d'une véritable donnée de comparaison inter-dispositif, il est nécessaire de les établir expérimentalement. Outre le fait que cette exigence est préjudiciable à la nature prédictive du modèle, comme nous l'évoquerons ci-après, l'obtention de ces valeurs, même expérimentalement, n'est pas toujours triviale.

Section III - Evaluations expérimentales

Comme nous venons de l'évoquer, les modèles prédictifs fournissent des outils de comparaison rapide dans certaines situations, normalement prédéterminées, pour lesquelles les paramètres relatifs au modèle utilisé ont dors et déjà été mesurés. En revanche, dans le cas contraire et général, ces paramètres doivent être obtenus par des expérimentations préalables.

Par ailleurs, les claviers logiciels ne sont plus uniquement de simples distributions de boutons statiques. La saisie de caractères est souvent complétée par des aides dynamiques, et contextualisées par les précédentes saisies (comme, le plus couramment, des listes de complétion). Les processus cognitifs relatifs à l'utilisation de ces aides dynamiques ne sont pas pris en charge par certains modèles (cas de Soukoreff-MacKenzie) ou sont difficiles à modéliser. Ils ont des impacts complexes à évaluer pouvant inférer non seulement sur les performances intrinsèques du clavier, mais aussi sur des performances à l'usage (en pouvant générer une fatigue cognitive, physique ou même oculaire par exemple).

Les modèles sont donc principalement exploités dans des cas simples comme la comparaison de dispositions de caractères ou comme une donnée d'étude parmi d'autres dans l'évaluation de claviers plus complexes faisant intervenir des arrangements dynamiques [Raynal 05a, Raynal 07c, Merlin 10a].

[4] Certaines sous-tâches, comme le pointage d'une touche virtuelle ou physique, répondent à la loi de Fitts, néanmoins les coefficients a et b restent relatifs au dispositif utilisé et aux caractéristiques psychomotrices de la population cible.

Pour une évaluation plus réaliste de ces systèmes complexes, il apparaît donc nécessaire de faire intervenir les utilisateurs finaux dans le cadre d'une démarche expérimentale. Plusieurs protocoles sont fréquemment utilisés. La tâche (III.1), les variables mesurées (III.2) et le mode de calcul des erreurs (III.3) varient en fonction des objectifs de l'évaluation.

III.1. Les principaux protocoles

III.1.a. La tâche à effectuer

Deux types de tâches sont généralement exploités.

En premier lieu, et de manière à évaluer l'efficacité d'un système de prédiction plutôt que l'efficacité global de la saisie, [Wester 03] propose, par exemple, une tâche de création de texte. Cette tâche permet effectivement d'évaluer la robustesse d'un système de prédiction en dehors d'un cadre prédéfini par l'expérimentateur. Le système est ainsi confronté au vocabulaire de l'utilisateur, aux éventuelles approximations orthographiques, etc.

Ce type de tâches pose en revanche de nombreux problèmes, exposés par [MacKenzie 02c], dans le cadre de l'évaluation des performances en elles-mêmes du système de saisie. Principalement la tâche de saisie est grevée par une seconde tâche dont il est difficile de mesurer l'impact. Il est impossible de discriminer si les latences, les hésitations ou encore les erreurs de l'utilisateur sont imputables au dispositif de la saisie ou bien aux aléas de la tâche de création. Par ailleurs, les caractéristiques du texte généré ne sont pas forcément représentatives de la langue (en termes de fréquence d'apparition des caractères ou de fréquences des bigrammes). Un résultat obtenu sur les performances de saisie via un artefact dans le cadre de l'expérimentation n'est pas forcément facilement étendu à la saisie de texte en général.

III.1.b. Présentation de l'informations

En conséquence, la tâche la plus couramment utilisée dans le cadre de l'évaluation des systèmes de prédiction est la tâche de recopie. Malgré tout, une indétermination subsiste sur le temps consacré à la saisie et le temps consacré à la lecture du texte. Pour lever cette ambiguïté, [Evreinova 04, Zhai 03] proposent la recopie de mots isolés permettant à l'utilisateur d'apprendre le mot avant d'initier la saisie. Le temps de saisie est pris en compte entre la frappe du premier caractère du mot et la frappe du dernier caractère (le temps de saisie du premier caractère est donc déconsidéré).

Bien que la tâche soit robuste en matière de calcul du temps, la saisie s'en trouve saccadée et moins naturelle par rapport à une activité réelle de saisie. C'est pourquoi [Zhai 01, Goodman 02, Raynal 05b] préfèrent, à la saisie de mots isolés, la saisie de phrases courtes. Un jeu de phrases courtes en anglais, respectant la fréquence d'apparition des caractères et des bigrammes, a été notamment créé et standardisé [MacKenzie 03] et régulièrement réutilisé [Isokoski 04b, Pavlovych 04].

Bien que la technique soit moins précise en termes de mesure du temps, d'autres projets [Matias 96, Ward 00, Ingmarsson 04] proposent d'évaluer la saisie à travers la recopie de textes plus longs. Cette recopie ne permet plus de distinguer le temps alloué à la lecture du temps concrètement alloué à la saisie, introduisant un biais dans l'estimation du temps de saisie. En revanche, elle permet d'observer l'évolution des performances permettant notamment de caractériser la fatigue engendrée par l'utilisation de l'artefact.

La manière de représenter le texte à recopier peut en outre aider à minimiser ce biais. Plusieurs protocoles [Magnien 04, Pavlovych 05] proposent ainsi d'intercaler la ligne à recopier et le texte déjà saisi permettant à l'utilisateur de localiser facilement le texte à saisir

et de constater rapidement, par comparaison entre les deux lignes, une éventuelle erreur. D'autres [Vigouroux 04, Ingmarsson 04] proposent d'améliorer cette technique en faisant défiler le texte. Le texte à recopier est ainsi toujours situé au même endroit (au milieu de l'écran).

A noter qu'une dernière alternative reste la dictée permettant de supprimer définitivement le temps dédié à la lecture. Mais, outre le fait que la solution est chronophage pour l'expérimentateur, elle est stressante pour l'utilisateur [Ward 00] qui se ressent indirectement questionné sur sa maîtrise de l'orthographe. La méthode introduit un nouveau biais sur l'interprétation des erreurs : erreur d'orthographe ou erreur de saisie.

III.2. Les variables généralement mesurées

III.2.a. Temps de saisie

Au cours des évaluations expérimentales, l'une des principales données mesurée est bien entendu le temps de saisie. Néanmoins en fonction du protocole expérimental, et particulièrement si l'expérimentation prévoit une ou plusieurs sessions, le sens, la représentativité du temps saisie évalué diffère.

Les expérimentations mono session [Magnien 04, Raynal 07a, Bi 10] comparent en général deux dispositifs voisins, l'un proposant une amélioration ou une aide supplémentaire par rapport à l'autre. Par exemple, dans la comparaison entre le clavier FishEye [Raynal 07a] et AZERTY, les deux claviers ont pour support la même distribution de touches et la même technique d'interaction. En revanche les performances sont légèrement optimisées sur FishEyeKeyboard dans la mesure où le pointage est facilité par l'accroissement des touches ciblées. De même, pour PoBox [Masui 99], la liste de prédiction est une assistance supplémentaire qui démarque deux claviers identiques.

Bien qu'elles soient parfois également comparatives, les expérimentations multi sessions visent plutôt à évaluer les performances d'un clavier par rapport à son usage à long terme [Bellman 98, McQueen 95, Zhai 02a], autrement dit, les performances que pourait obtenir un expert. Un nouveau concept, un nouveau clavier est en principe comparé à un clavier étalon représentatif des pratiques courantes des usagers (comme AZERTY ou QWERTY pour l'anglais, ou encore T9 pour des claviers de téléphones). Les caractéristiques peuvent également être établies de manière isolée à travers un protocole expérimental de référence.

Dans ce genre d'expérimentations, l'utilisateur est le plus souvent confronté à un nouveau dispositif proposant des caractéristiques, sinon nouvelles, au moins inhabituelles telles qu'une nouvelle distribution de touches [Zhai 00, Zhai 02a], une nouvelle technique d'interaction [Ward 00, Isokoski 04], ou même les deux [Zhai 03]. Le plus souvent, l'utilisateur déjà familiarisé avec les pratiques standards, est plus efficace avec le clavier de référence au cours des premiers usages. Les évaluations visent donc en premier lieu à démontrer l'existence d'un degré d'expertise (défini par MacKenzie comme le *crossover* [MacKenzie 99], cf. Figure 6) pour lequel l'utilisateur sera plus performant avec le nouveau clavier qu'avec son mode de saisie habituel, et en second lieu de définir les performances maximales accessibles avec le clavier. On peut noter que ces performances maximales sont rarement obtenues au cours de l'expérimentation. Sont ainsi retenues comme performances maximales les performances obtenues en expérimentation [McQueen 95], ou une projection de la progression des performances [Bellman 98].

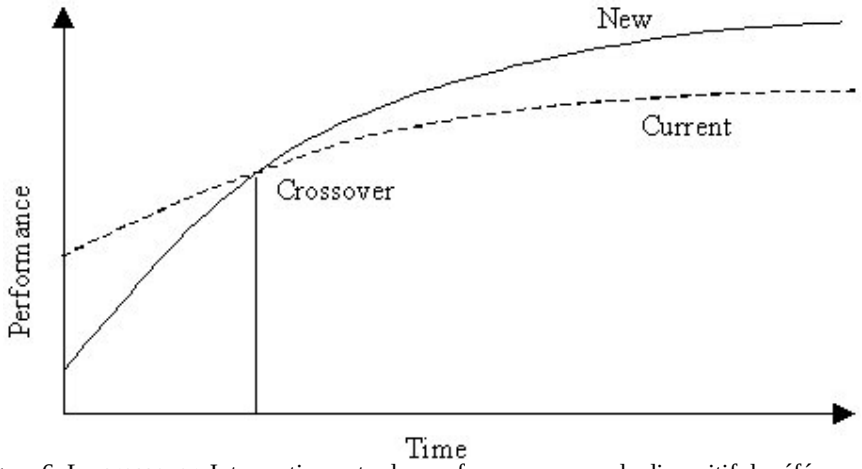


Figure 6. Le ***crossover,*** Intersection entre les performances avec le dispositif de référence et le dispositif expérimenté [MacKenzie 99]

III.2.b. Distance parcourue par le curseur

En quelques occasions, essentiellement lorsque l'artefact est destiné à un public d'utilisateurs handicapés moteurs des membres supérieurs, une autre donnée peut être mise en exergue : la distance parcourue par le pointeur. Cette donnée revêt un intérêt spécifique dans ces circonstances car le déplacement du pointeur est fatigant pour le public concerné.

Cette donnée doit néanmoins être relativisée dans la mesure où la réduction des efforts relative à la diminution des distances ne peut être contrebalancée par une exigence supplémentaire dans la précision du pointage. En conséquence, il ne s'agit pas de réduire les dimensions d'un artefact.

En revanche, à paradigmes d'interaction égaux et tailles de touches égales, un artefact permettant de réduire les distances parcourues par le pointeur peut donc éventuellement s'avérer plus performant, pour un utilisateur expert, même si le temps de saisie est au final supérieur.

III.2.c. Les erreurs

Comme nous l'avons implicitement évoqué, les erreurs et leur calcul peuvent être influencés par la nature de la tâche à exécuter et par la manière de présenter l'information (facilitant ou non la détection de l'erreur). Cependant, le calcul des erreurs peut être encore plus lourdement impacté par la stratégie de correction des erreurs. Ainsi, deux principales stratégies se distinguent en matière de correction des erreurs.

La première stratégie consiste à demander à l'utilisateur d'ignorer les erreurs. Les erreurs ne seront en conséquence pas corrigées dans le texte final produit. Ce nombre d'erreurs résultera du nombre d'insertions, de suppressions, de substitutions ou permutations effectuées par rapport au texte initial. Cette distance entre le texte produit et le texte de référence peut être obtenue par les algorithmes de Levenstein [Levenstein 66] ou de Kruskal [Kruskal 83a, Kruskal 83b]. Soukoreff et MacKenzie [Soukoreff 01, Soukoreff 04] en déduisent le taux d'erreur MSD (Minimum String Distance) exprimé par :

$$\boldsymbol{MSD} = \frac{\boldsymbol{MSD}(\boldsymbol{I},\boldsymbol{P})*100}{\max(|\boldsymbol{I}|,|\boldsymbol{P}|)}$$

où MSD (I, P) est la distance d'édition entre le texte initial I et le texte produit P, et |I| et |P| sont le nombre de caractères des textes I et P.

La seconde stratégie consiste à contraindre l'utilisateur à produire un texte final sans erreur. Dans certaines expérimentations, l'utilisateur est empêché de poursuivre la saisie tant que le caractère courant à saisir n'a pas été correctement saisi [Raynal 05a, Merlin 09a]. Dans ces circonstances, l'utilisateur ne corrige pas son erreur mais se doit de saisir le bon caractère pour pouvoir continuer. Dans d'autres expérimentations, l'utilisateur est conduit, par un feedback, à corriger son erreur [Soukoreff 01] mais il est toujours en mesure de saisir les caractères suivants. A noter qu'en fonction de ces deux cas, la séquence de frappe résultante peut s'avérer très différente.

Dans la mesure où le texte final est dépourvu d'erreur, la distance entre les deux textes est nécessairement de 0. Dans ce contexte, Soukoreff et MacKenzie [Soukoreff 01, MacKenzie 02b] proposent d'évaluer le taux d'erreur en fonction du nombre moyen de frappes nécessaires pour saisir un caractère (KSPC – Key Stroke Per Character). Le taux d'erreur KSPC s'exprime par :

$$\mathrm{KSPC} = \frac{\boldsymbol{Nf}}{|\boldsymbol{T}|}$$

où Nf est le nombre total de frappes occasionnées par la saisie du texte, par les erreurs et éventuellement par les saisies nécessaires à leur correction, et |T| le nombre de caractères contenus dans le texte original.

Une stratégie intermédiaire [Soukoreff 03] consiste à laisser l'utilisateur libre de corriger les erreurs qu'il perçoit. Dans ces circonstances, le texte produit est partiellement corrigé. La séquence créée pour engendrer le texte contient des caractères correctement saisis (noté C - *Correct*), des caractères erronés et non corrigés (notés INF – *incorrect and not fixed*), des caractères erronés mais corrigés (IF – *Incorrect but fixed*) et enfin des caractères saisis, *backspace* par exemple, pour corriger des erreurs (F – *Fixed*). Le taux d'erreur dépend donc à la fois des erreurs corrigées (KSPC) et des erreurs non corrigées avec :

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

Et

$$\mathrm{M}\boldsymbol{SD} = \frac{\boldsymbol{INF} * 100}{\boldsymbol{C} + \boldsymbol{INF}}$$

L'avantage de la première stratégie est que, n'obligeant pas l'utilisateur à corriger ses erreurs, le nombre de caractères saisis reste voisin du nombre de caractères contenus dans le texte facilitant ainsi : l'estimation du temps de saisie par caractère et la comparaison des résultats avec les résultats théoriques. En revanche, la non-correction des erreurs contrarie le réflexe naturel de beaucoup d'utilisateurs.

La seconde stratégie correspond à une tâche plus naturelle et plus fidèle à la réalité de la saisie de texte, mais le temps de correction de l'erreur influe sur le temps global de saisie engendrant évidemment des saisies supplémentaires. Par ailleurs, la manière de corriger les

erreurs[5] modifie considérablement le résultat obtenu en termes de calcul des erreurs et en termes de temps de saisie.

Les différences significatives dans les protocoles expérimentaux réduisent l'objectivité des temps de saisie obtenus mais compliquent surtout l'interprétation des erreurs. De fait, lors des revues comparatives de clavier [MacKenzie 99, Zhai 02a] le taux d'erreur engendré par le clavier n'est pas cité.

Section IV - Synthèse : moyens différents mais finalité commune : évaluer les performances d'un expert

En conséquence, les principaux modèles prédictifs proposés permettent, dans des cas plus ou moins bien délimités et avec des degrés d'approximation divers, d'évaluer les performances théoriques d'un utilisateur expert. Bien que le modèle de Soukoreff et MacKenzie cible également les performances d'un novice, de l'aveu même des auteurs [Soukoreff 95], opinion corroborée par d'autres études [Pavlovych 04, Das 08, Merlin 10b], les valeurs obtenues sont trop pessimistes pour être représentatives.

Parallèlement, hormis les situations où l'expérimentation ne se prononce pas sur les performances intrinsèques du clavier (cas des études comparatives avec un petit nombre de sessions), les évaluations visent également à dégager les performances maximales atteignables par un utilisateur de fait expérimenté. Le taux d'erreurs n'étant pas toujours facilement interprétable et surtout rarement comparable entre deux expérimentations distinctes, il est rarement référencé. De même, il est rarement fait état des performances minimales[6] ou du nombre de sessions nécessaires pour atteindre le crossover (Figure 6). Les efforts pour prendre d'autres paramètres en lignes de compte, tels que la charge mentale relative à l'utilisation du clavier [Poirier 07], sont généralement ignorés.

La référence qui subsiste donc comme valeur représentative des caractéristiques d'un clavier est la vitesse maximale de saisie pour un utilisateur expert (ou éventuellement une diminution en termes de distance parcourue par le dispositif de pointage si le public concerné est un public d'utilisateurs handicapés moteurs). Pourtant, comme nous allons l'observer dans le prochain chapitre, les performances d'un expert restent une donnée très relative.

[5] Par exemple : afin de corriger l'erreur, est-ce que l'utilisateur doit éliminer tous les caractères saisis après celle-ci avant de l'atteindre ou est-ce qu'il peut directement sélectionner la zone à corriger et substituer l'erreur par une nouvelle séquence de caractères ?

[6] [Soukoreff 95] propose un récapitulatif comparatif incluant les performances minimums et maximums de différents claviers. Néanmoins, ce récapitulatif vise à comparer différentes techniques d'évaluation et n'a pas pour objectif de comparer à proprement parler les claviers. De fait, les valeurs en elles-mêmes sont obtenues par des techniques différentes et sont peu comparables.

Chapitre 2 - Questionnement sur la notion d'expert

Une vidéo de présentation[7] décrit TouchPal comme un clavier susceptible d'atteindre une vitesse de saisie de 72 wpm (*words per minute*) pour un utilisateur expert. Cette vitesse est exceptionnellement élevée pour un clavier logiciel dans la mesure où cela correspond à environ 6 caractères saisis par seconde. A titre de comparaison, le clavier AZERTY est, par exemple, évalué à environ 28 wpm [Zhai 02a] pour un usage via un stylet sur un écran tactile.

TouchPal est néanmoins pourvu d'une liste de complétion permettant fréquemment de saisir en une seule frappe la fin des mots. Même ainsi, la valeur de 72 wpm suscite une admiration interrogative. La vidéo fait pourtant la démonstration de cette performance : l'utilisateur y fait un usage intensif de la liste de complétion avec une anticipation parfaite de son contenu. L'utilisateur est non seulement expert de l'usage du clavier, mais il est également expert de la phrase saisie dans le cadre de la démonstration, au point qu'il anticipe avec perfection les mots et l'ordre des mots dans la liste de complétion. L'usage de la liste ne requiert aucun coût cognitif. A priori cette performance est donc reproductible après entraînement sur un corpus limité à quelques phrases. En revanche, les performances du même utilisateur prises sur une phrase quelconque doivent normalement chuter considérablement (à moins que celui-ci connaisse l'ordre des résultats du système de prédiction pour tous les mots du dictionnaire). 72 wpm ne représentent donc pas les performances d'un utilisateur « expert », mais les performances d'un utilisateur « parfait ».

A travers les travaux réalisés dans le contexte de KeyGlass, nous observerons les limites de l'application aux claviers dits complexes[8] des modèles théoriques modélisant les performances d'un utilisateur expert. Puis, nous discuterons de la notion même d'expertise et de la problématique de l'expertise de l'utilisateur dans le cas des évaluations expérimentales comparatives.

Section I - Modélisation, simulation et évaluation théorique versus résultats expérimentaux

Dans le cadre de son étude, Mathieu Raynal [Raynal 05a] a été conduit à réaliser une évaluation théorique puis expérimentale du système KeyGlass, système basé sur une assistance relative à un système de prédiction. Cette évaluation se révèle illustrative des limites de l'application des modèles théoriques sur les claviers complexes. Nous avons par ailleurs confirmé ces observations dans le contexte de l'évaluation du clavier SpreadKey [Merlin 09a, Merlin 10a] également évoqué dans le chapitre Chapitre 10 - de cette thèse.

Nous évoquerons en conséquence les apports du système KeyGlass et les résultats des évaluations illustrant une différence très significative entre ceux attendus du point de vue théorique et ceux obtenus expérimentalement.

[7] http://www.cootek.com/

[8] Claviers proposant des aides dynamiques, des modes d'interaction alternatifs, etc. et dont le mode de fonctionnement diffère d'une simple pression de touche pour effectuer la saisie d'un unique caractère

I.1. KeyGlass

KeyGlass s'inspire du clavier logiciel augmenté proposé par Isokoski [Isokoski 04] (cf.Figure 7). Isokoski observe que, dans la langue finlandaise (ce qui est vrai pour de nombreuses autres langues et entre autres l'anglais ou le français), très fréquemment une voyelle succède à une consonne. Par ailleurs, les mots ayant une longueur moyenne de 5 caractères, dans environ un cinquième des cas un caractère est suivi d'un espace (un peu moins en réalité car un mot peut également être suivi d'un caractère de ponctuation). Il propose en conséquence d'effectuer la saisie d'une consonne puis d'une voyelle ou d'un espace en une seule interaction. Sur le clavier logiciel manipulé par l'intermédiaire d'un stylet, lorsque l'utilisateur pointe un caractère, il a dans la continuité la possibilité de saisir un second caractère par marking menu [Kurtenbach 91]. Le Pie-menu [Hopkins 91] support de l'interaction de marking-menu contient en conséquence les voyelles (ainsi que le caractère espace).

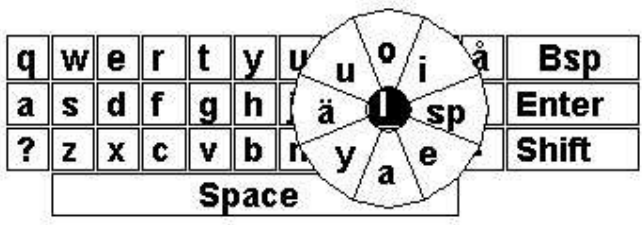


Figure 7. Clavier augmenté par un marking menu [Isokoski 04]

KeyGlass se destine prioritairement à des utilisateurs handicapés moteurs des membres supérieurs. Les utilisateurs de KeyGlass sont en mesure d'interagir avec un clavier logiciel via un dispositif d'interaction tel qu'un trackball. Néanmoins le travail de pointage reste une tâche laborieuse et, plus la distance à parcourir par le pointeur est longue, plus la tâche est fatigante.

L'intérêt du concept proposé par Isokoski, dans le cadre de l'utilisation d'un clavier logiciel par des utilisateurs handicapés moteurs, est qu'il permet de rapprocher des caractères fréquemment saisis de la dernière frappe effectuée, réduisant ainsi la distance à parcourir pour effectuer ces frappes. Néanmoins, l'interaction proposée par Isokoski est inappropriée pour des utilisateurs handicapés : la tâche de marking-menu est exigeante du point de vue moteur. En outre un simple pie-menu masquerait les autres caractères du clavier. Par ailleurs, de manière à permettre l'apprentissage du marking-menu, le menu propose systématiquement les mêmes caractères. En conséquence, dans de très nombreux cas, et notamment lorsqu'il est nécessaire de saisir une consonne, le menu est inutilisable.

Raynal [Raynal 05a] observe, en outre, qu'en fonction des caractères préalablement saisis, les 4 caractères les plus probables, relatifs à la fréquence des mots dans la langue française, regroupent statistiquement 80% des chances d'être effectivement saisis[9]. En conséquence, après chaque saisie d'un caractère, KeyGlass ouvre un « pie-menu » ou plus exactement une toolglass [Bier 93] contenant systématiquement les quatre caractères les plus probables. Les caractères sont déduits d'un système de prédiction basé sur les caractéristiques de la langue de l'utilisateur. Cependant, de manière à ne pas occulter une partie du clavier, les nouvelles touches sont disposées à l'intersection entre les touches principales du clavier (cf. Figure 8).

[9] Le pourcentage est sensiblement équivalent pour l'anglais

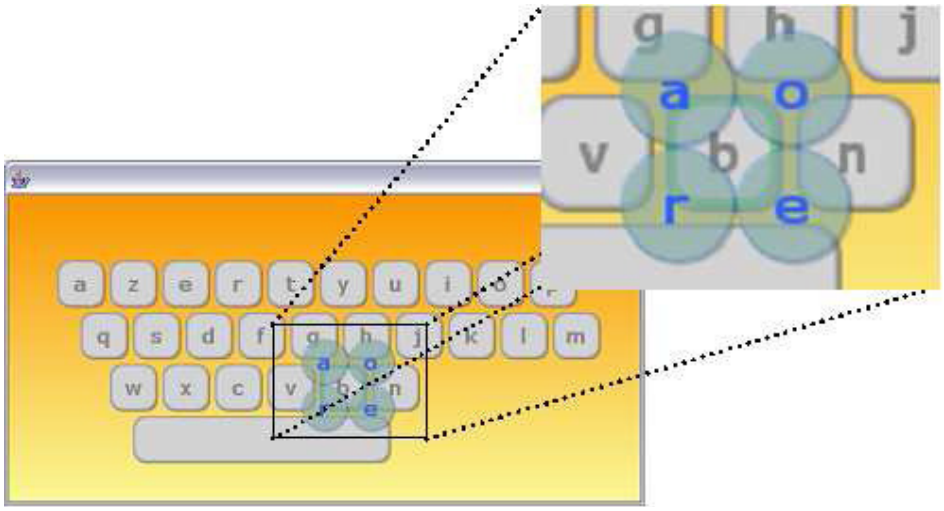


Figure 8. KeyGlass

I.2. L'expertise dans l'évaluation théorique de KeyGlass

L'évaluation théorique proposée par Raynal [Raynal 05a] repose sur le modèle de Soukoreff et MacKenzie. Néanmoins, en raison de l'évolution dynamique du clavier, résultant des touches additionnelles dont la position et le caractère représentés sont relatifs aux précédentes saisies, une évaluation statique du clavier (comme évoquée en section II.1) n'est pas possible. Raynal propose, en conséquence, d'effectuer une simulation de la saisie et de mesurer le temps de saisie de chaque caractère en fonction de chaque nouveau contexte.

Les mots saisis au cours de la simulation sont les mots contenus dans le dictionnaire du scrabble. Le poids du temps de saisie est pondéré par la fréquence du mot dans la langue française.

Dans la mesure où les touches additionnelles proposent un caractère redondant par rapport à un caractère déjà présent sur le clavier, lorsque la simulation doit mesurer le temps de saisie de l'un des quatre caractères présents parmi les touches additionnelles, deux possibilités peuvent être prises en compte : soit le temps de saisie en ciblant la touche additionnelle, soit le temps de saisie du caractère normal. Dans la simulation, Raynal va considérer une utilisation maximale des touches additionnelles et donc prendre en compte préférentiellement le temps d'accès à ces touches lorsque le caractère ciblé est présent parmi elles.

En revanche, il va considérer trois niveaux d'expertise différents :

- Un niveau d'expertise « novice » : l'utilisateur n'a connaissance ni de la disposition originelle des touches du clavier (AZERTY), ni d'anticipation sur le résultat fourni par le système KeyGlass. En conséquence, au temps de pointage il faut additionner le temps de recherche visuelle de la touche en fonction des deux circonstances suivantes : le caractère ciblé est présent ou non parmi les touches additionnelles. On considère que l'utilisateur recherche en premier lieu parmi les touches additionnelles (recherche parmi 4 caractères), puis dans le reste du clavier si le caractère n'a pas été rencontré au sein des touches additionnelles (recherche parmi les 27 caractères du clavier).

- Un niveau où l'utilisateur est « expert » du clavier AZERTY, en conséquence la tâche est grevée par la recherche du caractère parmi les 4 touches additionnelles. L'utilisateur

consacre un temps à l'analyse des 4 caractères proposés. Si le caractère est présent parmi les quatre touches proposées, l'utilisateur va sélectionner ce caractère, sinon, il va sélectionner la touche contenant normalement le caractère sans exiger que cette touche soit recherchée visuellement.

- Un niveau ou l'utilisateur est « parfait », au sens ou nous l'avons exprimé pour TouchPal : non seulement l'utilisateur est expert du clavier AZERTY, mais il est également en mesure d'anticiper le résultat de la prédiction. On néglige dès lors les coûts de recherche d'une touche et seul le temps de pointage est considéré en ciblant de façon optimale : soit une touche additionnelle si le caractère est présent parmi elle ; soit, dans le cas contraire, la touche contenant normalement le caractère sur le clavier AZERTY.

La simulation va permettre de comparer les performances théoriques de ces différentes classes d'utilisateurs. Les valeurs utilisées en paramètre des lois de Fitts et Hick-Hyman, correspondant aux valeurs utilisées par Soukoreff et MacKenzie [Soukoreff 95], simulent des utilisateurs normaux (sans déficience motrice) de manière à faciliter la comparaison avec des données expérimentales.

Les résultats, présentés ci-dessous (Tableau 2), illustrent des perspectives extrêmement diverses en fonction du degré d'expertise de l'utilisateur. En fournissant une sélection des caractères les plus probables, KeyGlass devrait ainsi être susceptible d'améliorer les performances d'un utilisateur complètement novice d'environ 22%.

En revanche, pour un utilisateur expert de la distribution de touche AZERTY, la réduction des distances parcourues par le pointeur grâce au système KeyGlass ne permet pas de compenser le temps de recherche parmi les 4 caractères proposés dynamiquement. L'usage systématique des touches engendrerait ainsi une perte d'efficacité d'environ 40%.

Pour pouvoir profiter du dispositif en termes de réduction de temps de saisie, un utilisateur devrait donc être en mesure de pouvoir anticiper au moins une partie des résultats du système de prédiction. Dans ces circonstances uniquement, un gain de temps serait permis avec même un gain théorique de 76% pour un utilisateur « parfait », capable d'anticiper systématiquement le résultat du système de prédiction.

	AZERTY	**KeyGlass**	
Novice	5,57 wpm	6,81 wpm	+ 22%
Expert	18,35 wpm	10,92 wpm	- 40%
Parfait	18,35 wpm	32,26 wpm	+ 76%

Tableau 2. Comparaison des performances entre KeyGlass et AZERTY pour différents niveaux d'expertise

Néanmoins, ces résultats sont établis en cas d'un usage systématique des touches proposées. Rien n'empêcherait par exemple à un utilisateur « expert » avec le clavier AZERTY et novice avec KeyGlass de n'exploiter les touches additionnelles qu'un pourcentage limité de fois, voire de ne pas les utiliser du tout. En conséquence, il est difficile de préjuger des performances exactes d'un utilisateur expert, celles-ci pouvant ainsi osciller de 60% à 100% par rapport aux performances obtenues avec AZERTY.

De même, s'il est peu réaliste d'envisager l'existence d'un utilisateur « parfait ». Il est difficile d'envisager ce qu'un utilisateur sera concrètement capable d'apprendre et les touches

additionnelles qu'il sera en mesure d'anticiper à long terme. En conséquence, les performances d'un utilisateur « aguerri » au système KeyGlass devraient osciller entre celles d'un utilisateur expert (11wpm, 60% des performances du clavier AZERTY) et celle d'un utilisateur « parfait » (32wpm, 170% des performances du clavier AZERTY), soit du simple au triple. Autrement dit, les seules lois modélisant le pointage et la recherche visuelle d'une touche ne permettent pas a priori de prédire les performances d'un utilisateur avec le système KeyGlass (en dehors éventuellement de celles d'un utilisateur complètement novice à la fois avec le clavier AZERTY et avec le système KeyGlass).

I.3. L'expertise dans l'évaluation expérimentale

Une évaluation expérimentale est venue compléter l'évaluation théorique de KeyGlass. Cette expérimentation a été effectuée auprès de 6 sujets valides et auprès de 6 sujets handicapés. Elle engageait les utilisateurs dans 20 sessions contenant deux exercices de recopie des mêmes 30 mots. Un exercice était réalisé avec le clavier AZERTY, le second avec l'addition du système KeyGlass. L'ordre des exercices était alterné à chaque session pour contrebalancer les effets d'apprentissage.

Les résultats, restitués dans les figures suivantes, présentent en rouge les valeurs relatives aux performances des usagers valides, en vert les valeurs des usagers handicapés, et en bleu la moyenne globale.

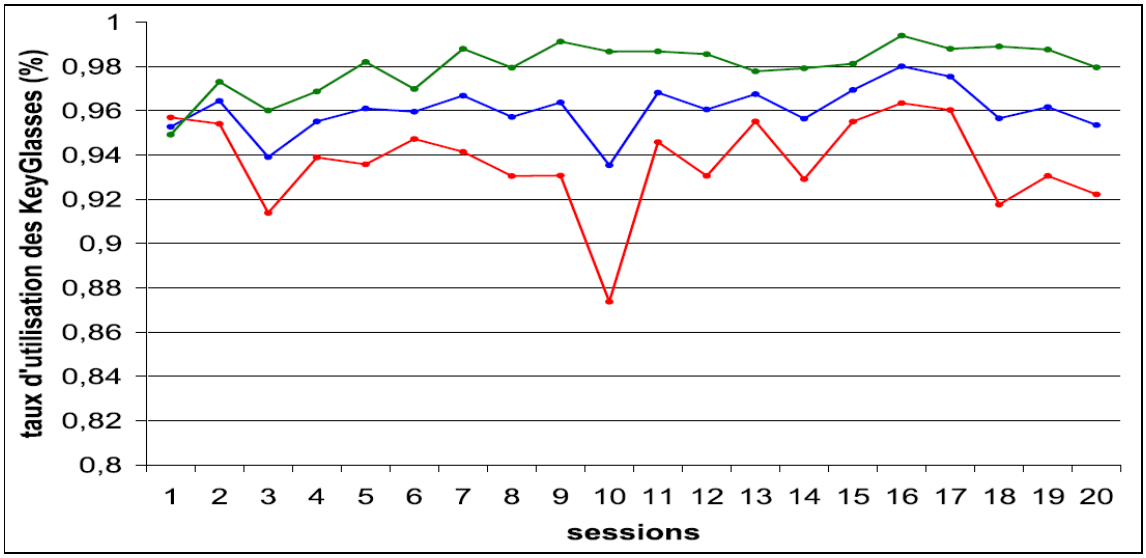


Figure 9.Taux d'utilisation des KeyGlass

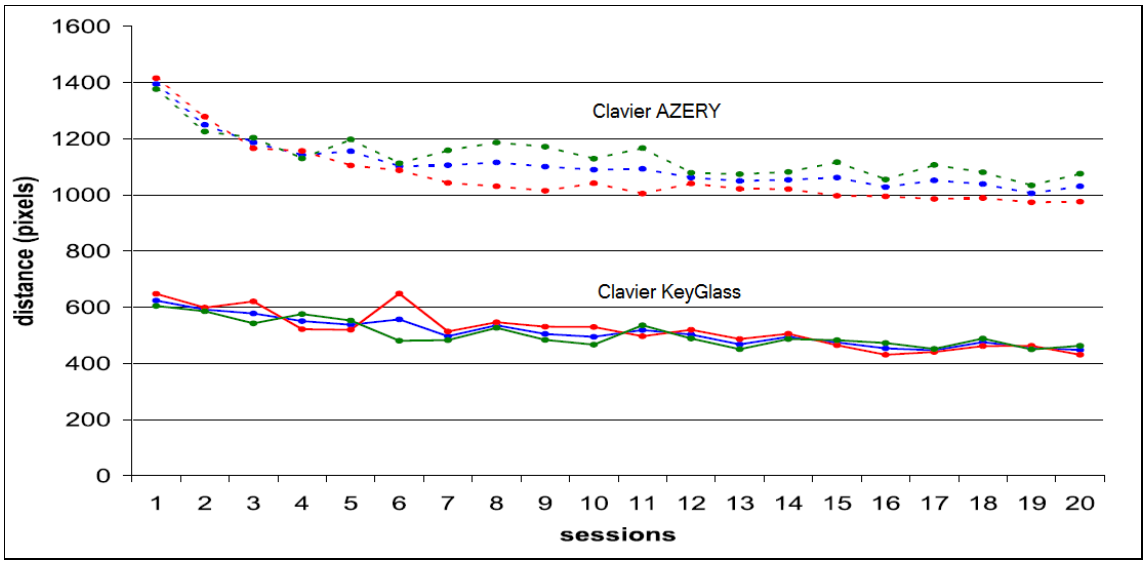


Figure 10. Distance par mot parcourue par le curseur

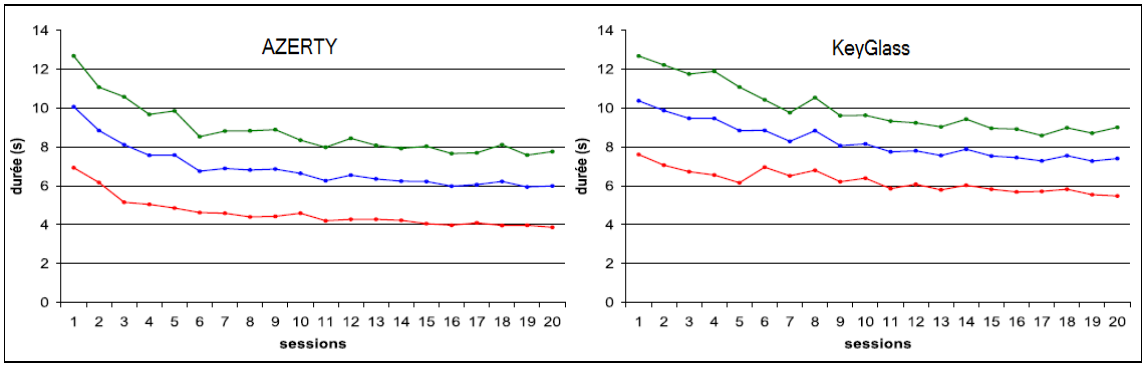


Figure 11. Temps de saisie par mot

Les résultats illustrent un usage relativement intensif (cf. Figure 9) du système KeyGlass. Ce taux d'utilisation est élevé quelque soit la population cible, et ce quelque soit le degré d'expertise de l'utilisateur. Le système reste néanmoins légèrement plus utilisé par les utilisateurs handicapés, ceci étant justifiable par le fait qu'ils sont les plus bénéficiaires du système. En effet, ces utilisateurs ayant un temps de pointage plus long, le coût nécessaire à l'analyse des touches additionnelles est proportionnellement moins élevé pour eux. En outre, comme nous pouvons l'observer Figure 10, les distances parcourues par le pointeur, considérablement réduites grâce au système, procurent un confort supplémentaire pour les utilisateurs handicapés.

En revanche, nous pouvons observer que l'incertitude sur la capacité à anticiper les résultats du système de prédiction est vérifiée. Malgré, la conservation d'un vocabulaire constant pour conduire l'expérimentation, l'utilisateur n'a pas été en mesure d'anticiper ces résultats. En conséquence, les performances, du point de vue de la vitesse de saisie (cf. Figure 11), sont restées inférieures aux performances obtenues avec le simple clavier AZERTY.

Section II - Les limites de la notion d'expertise

Les expérimentations destinées à évaluer des claviers « simples » [Raynal 05b, Zhai 00] (dont seule la distribution des caractères est altérée), prennent en compte, comme critère

d’expertise, le nombre de sessions réalisées au moyen du dispositif. Les mêmes phrases sont saisies par l’utilisateur au cours des différentes sessions jusqu’à ce que leur saisie devienne automatique et que la courbe de progression de l’utilisateur soit stabilisée.

Tout d’abord, comme nous l’avons évoqué précédemment, ce même protocole ne peut être utilisé avec pertinence pour l’évaluation de claviers utilisant un système de prédiction. Le protocole conduirait à évaluer non pas les performances d’un utilisateur expert, caractéristiques d’un usager quotidien, mais les performances d’un utilisateur « parfait » qui ne seront pas représentatives de l’usage réel du clavier. Le corpus de textes utilisé au cours des différentes sessions peut donc influer considérablement sur l’apprentissage et sur le résultat de l’évaluation. Tel ou tel corpus pourrait en outre favoriser un dispositif en particulier compliquant la comparaison de différents dispositifs.

Par ailleurs, la caractéristique « expert » est supposée représentative des performances d’un usager quotidien : on considère qu’un utilisateur expert ayant acquis les automatismes suffisants maintient ces performances à travers un usage régulier du clavier. Cependant, dans le cadre de l’usage d’un clavier muni d’un système de prédiction, l’usager n’est pas en mesure d’acquérir tous les automatismes dans toutes les situations. Outre le nombre d’utilisations, la fréquence d’utilisation pourrait également influer significativement sur les performances. [James 01] observe par exemple qu'un utilisateur régulier d'un téléphone portable saisissait, en 2001, environ 20 messages brefs (SMS) par mois, ce qui ne lui permettait pas d'avoir une grande anticipation relativement au système de prédiction. Dans ces circonstances, l'anticipation était possible uniquement pour les mots les plus usuels.

Face à ces claviers, il existe donc plusieurs profils d’experts et il serait pertinent d’en envisager plusieurs dans le cadre de la comparaison des claviers. En effet, certains dispositifs, par exemple comparables aux autres pour un usage fréquent, pourraient également offrir des outils mnémotechniques permettant de soutenir les performances dans le cas d’un usage moins fréquent et s’avérer donc globalement plus efficaces.

Section III - Synthèse

Ainsi, comme nous avons pu l’observer dans le cadre des travaux cités, si après un certain temps d’apprentissage nous pouvons nous abstraire des coûts cognitifs engendrés par l’usage d’un clavier simple, en revanche, dès qu’un clavier fait intervenir des caractéristiques dynamiques (et notamment relatives au contexte linguistique) ces coûts cognitifs ne peuvent plus être négligés.

En premier lieu, ces caractéristiques ne permettent plus de modéliser avec précision le comportement d’un utilisateur expert au moyen des lois prédictives telles que KLM, Fitts et Hick-Hyman, etc.

En second lieu, elles ne permettent plus de faire émerger un seul profil d’utilisateur expert, mais plusieurs, dont les performances seront sensibles à la fréquence d’utilisation de l’artefact.

Enfin, étant donné la sensibilité des claviers logiciels au corpus de textes utilisés pendant l’expérimentation, il est difficile de définir une méthode objective comparant ces différents niveaux d’expertise entre différents artefacts.

Les performances d’un utilisateur expert ne peuvent donc plus être considérées comme une caractéristique absolue de comparaison des claviers logiciels.

Chapitre 3 - Questionnement sur la notion de novice

Par ailleurs, en ne considérant que des claviers « simples », de nombreux projets ont proposé de nouvelles solutions de claviers logiciels, partant de la constatation que des claviers logiciels tels que les mini-AZERTY classiques [Goldberg 93, Zhai 00] (déjà théoriquement sous optimaux pour un usage à dix doigts), ou les claviers ambigus type T9 avec agencement alphabétique [Foulds 87, Levine 87, James 01], n'étaient pas les mieux adaptés à un usage tactile. Ces projets ont souvent affiché des performances théoriques (ou expérimentales) pour des utilisateurs experts, nettement supérieures aux performances théoriques des claviers communément utilisés [Arnott 92, Lesher 98]. En conséquence, on pourrait s'étonner que des claviers comme GAG [Raynal 05b] ou Metropolis [Zhai 00] (cf. Figure 12), dont l'usage par des utilisateurs experts est évalué à 55wpm (words per minute) et 51wpm contre moins de la moitié pour l'AZERTY ou le QWERTY ne soient pas plus répandus.

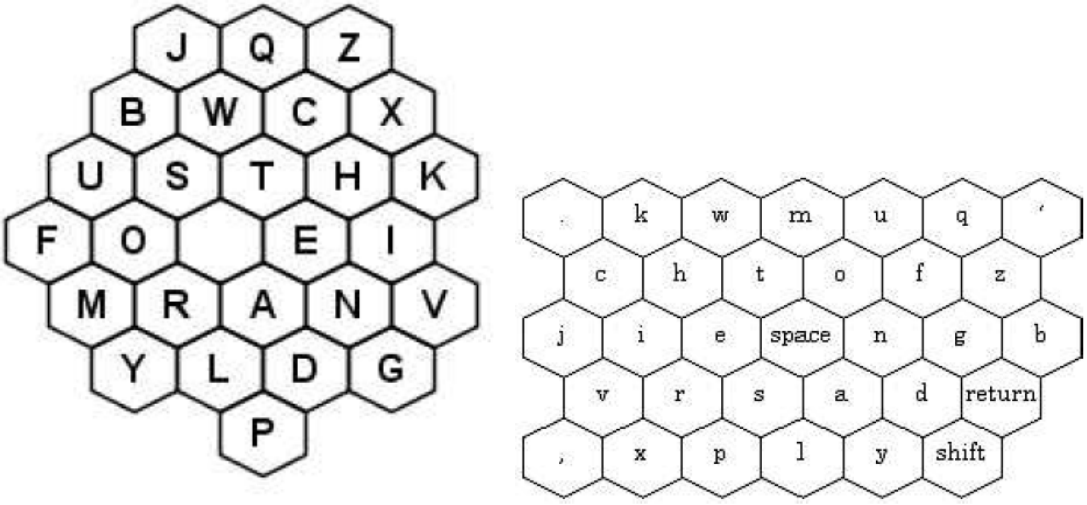


Figure 12. Claviers GAG (à gauche) et Metropolis (à droite)

Mais, si les projets s'attachent plus naturellement à démontrer la compétitivité et le bénéfice maximum des nouveaux outils et concepts proposés envisageables à long terme, cette seule donnée ne permet pas de qualifier les perspectives d'usage et d'appropriation par un usager. Dans un univers non contrôlé, dans le contexte d'un usage libre et hors cadre expérimental, un utilisateur va persévérer dans l'usage d'un outil si cet outil lui donne satisfaction à court terme. Ainsi, un bénéfice à long terme suggère indirectement un usage à long terme et consécutivement une bonne acceptation à court terme. En d'autres termes, plus le bénéfice de l'usage est à long terme, plus la tendance naturelle du concepteur est de se focaliser sur le long terme et plus la clef du problème se situe finalement à court terme.

Un premier élément caractéristique de l'utilisabilité d'un outil à court terme réside dans les performances de l'utilisateur au cours des premiers usages, les performances d'un novice. Pourtant, comme nous l'avons préalablement évoqué, les outils pour mesurer et comparer ces performances sont mal établis.

A l'instar des travaux menés par Das, Stuerzlinger et Pavlovych [Das 08, Pavlovych 04], nos premiers travaux nous ont conduits à évaluer la possibilité d'améliorer les modèles prédictifs de manière à rendre plus pertinent le calcul des performances d'un utilisateur novice. Le modèle que nous proposons ne vise plus à considérer l'utilisateur comme un « novice » dépourvu de connaissances, mais comme un utilisateur « débutant » avec le clavier et héritant d'une expérience passée facilitant éventuellement sa découverte d'un nouveau clavier.

Nous présenterons, en premier lieu, les principaux travaux visant à prendre en compte, dans les modèles prédictifs, l'apprentissage du clavier par un utilisateur (I). En second lieu, nous présenterons les claviers DashKey et MessagEase, supports de notre étude, puis nous montrerons comment nous proposons d'intégrer les acquis de l'utilisateur dans les modèles prédictifs afin d'améliorer la représentativité des résultats obtenus pour les performances d'un utilisateur débutant (II). Nous comparerons les résultats obtenus avec les valeurs expérimentales et discuterons les intérêts et limites de cette démarche (III).

Section I - Amélioration des modèles prédictifs

I.1. Temps de recherche versus temps de sélection

Le premier questionnement relatif à la modélisation des performances d'un utilisateur novice est relatif à l'utilisation de la loi de Hick-Hyman (cf. Chapitre 1 - Section I - I.3) pour établir le temps de recherche d'une touche.

Comme l'observent par exemple [Sears 01] ou encore [Pavlovych 04, Das 08], la loi suggérée par Hick-Hyman modélise un processus de recherche dichotomique, de recherche binaire, d'une cible visuelle. Elle est définie pour une tâche où l'utilisateur est capable de déterminer visuellement si l'élément recherché se situe dans un des deux sous-groupes de l'ensemble des cibles, puis dans l'un des deux sous-groupes du sous-groupe considéré jusqu'à ce que le sous-groupe ne contienne plus que l'élément recherché. Elle est donc définie pour une tâche de « complexité » logarithmique. En conséquence, la loi s'applique lorsqu'un utilisateur doit, par exemple, rechercher un item dans un menu ordonné alphabétiquement [Landauer 85].

En revanche, Sears argumente que, dans le cas de la recherche d'une touche sur un clavier par un utilisateur novice, le processus de recherche n'est pas dichotomique (dans la mesure où l'utilisateur n'a pas d'heuristique pour rechercher un élément dans l'une où l'autre de deux sous-parties du clavier), mais linéaire. Selon Sears, la loi de Hick-Hyman ne s'applique donc pas dans ce contexte.

En conséquence, Das [Das 08] propose de considérer deux types de tâches dans le cadre de la saisie de texte : la tâche linéaire de **recherche** d'une cible par un utilisateur novice, et une tâche logarithmique de **sélection** d'une cible par un utilisateur expert.

I.2. Modèle pour utilisateur novice

Pavlovych [Pavlovych 04] s'inspire à la fois du modèle de Soukoreff et MacKenzie adapté pour un usage à deux pouces via un clavier de téléphone portable [Silfverberg 00], et à la fois du modèle KLM adapté par Dunlop [Dunlop 99] pour une évaluation des performances d'un téléphone muni d'un système de prédiction. Sur la base de ces deux modèles, il propose un nouveau modèle permettant de décrire le comportement d'un novice face à un clavier de téléphone.

Il observe que, contrairement à ce que propose le modèle de Soukoreff et MacKenzie, les temps de latence entre les déplacements du pointeur pour un utilisateur novice ne sont pas uniquement dédiés à la recherche visuelle d'un caractère. Il décompose donc la tâche de saisie en plusieurs activités :

- Désignation de la cible ;
- Temps de recherche visuelle de la cible ;

Mais aussi :

- Réflexion concernant le caractère devant être saisi ;
- Réflexion relative au nombre de pressions qui doivent être effectuées sur la touche pour permettre la saisie du caractère considéré (pour des clavier ambigus, dans le cadre d'une technique de saisie telle que MultiTap par exemple)
- Lecture de la phrase destinée à être saisie (dans le cas d'une tâche de recopie) ;
- Vérification du résultat de la saisie.

Le temps nécessaire à l'exécution de chacune des sous-tâches est évalué expérimentalement, sauf la tâche de désignation de la cible dont le temps est calculé par l'intermédiaire la loi de Fitts.

I.3. Modélisation des effets d'apprentissage

Comme nous l'avons préalablement évoqué, Das [Das 08] corrobore les observations de Sears [Sears 01] sur la différence entre les temps de recherche d'une cible par un utilisateur novice[10] et le temps de sélection d'une cible par un utilisateur expert. Il propose de modéliser cette seconde par la loi de Hick-Hyman, qui correspondrait, de son point de vue, au temps nécessaire à la décision, à la sélection d'une cible par un utilisateur expert connaissant la répartition des touches sur le clavier.

Il propose donc un modèle destiné à refléter l'apprentissage en considérant que l'utilisateur va progressivement passer d'une stratégie de recherche de la cible à une stratégie de sélection de la cible.

Fondé sur le précédent modèle, il décompose le temps d'exécution de la tâche en : un temps de pointage qui n'évolue pas au cours de l'apprentissage ; un temps de recherche visuelle de la cible qui va être progressivement remplacé par un temps de sélection de la cible et des coûts cognitifs annexes qui vont progressivement décroître au cours de l'apprentissage.

Sur la base des expérimentations réalisées par Pavlovych [Pavlovych 04], il suggère un temps de 680ms pour le temps de recherche d'un caractère parmi les 26 caractères de l'alphabet latin répartis sur un clavier téléphonique, et un temps de 999ms pour les coûts cognitifs annexes. Le temps de sélection de la cible par un utilisateur expert est calculé sur la base de la loi de Hick-Hyman. Considérant que les caractères sont répartis sur 8 touches utiles, Pavlovych considère la loi de Hick-Hyman pour la sélection d'un item parmi 8. L'utilisateur étant désigné expert, il choisit comme coefficient b=1/7 (cf. Chapitre 1 - I.3). Le temps de sélection TS résultant est donc établi par :

[10] La tâche ne correspondant pas à une tâche théoriquement modélisée par la loi de Hych-Hyman

$$\boldsymbol{TS} = \frac{1}{7} \times \log_2(8) = 429\boldsymbol{ms}$$

Le modèle proposé est séduisant dans la mesure où les résultats se rapprochent des résultats obtenus expérimentallement sur un clavier téléphone, malheureusement, lorsqu'on l'applique à des claviers autres que les claviers ambigus, il en résulte quelques incohérences.

Établissons le modèle par exemple pour un simple clavier AZERTY. Comme le cite explicitement Das, les travaux de Sears ont montré que le temps de recherche d'un caractère est légèrement supérieur lorsque l'on place plusieurs caractères par touches. Le temps de **recherche** d'un caractère sur un clavier AZERTY devrait donc être inférieur à 680ms. En revanche, puisque nous disposons cette fois-ci de 27 touches utiles au lieu des 8 pour le clavier téléphone, le temps de **sélection** par un utilisateur expert est établi par 1/7 * log2 (27) soit 680ms ! Le temps de **sélection** pour un utilisateur expert devient donc supérieur au temps de **recherche** de la cible par un utilisateur novice.

Plusieurs hypothèses peuvent expliquer cette erreur. En premier lieu, le modèle de Soukoreff et MacKenzie, de même que le modèle de Das, supposent que la tâche de recherche visuelle de la touche, de sélection de la touche dans le cas du modèle de Das et la tâche de pointage sont effectuées successivement l'une après l'autre. Une première hypothèse est qu'une partie des deux tâches (pointage et recherche visuelle du caractère suivant ou pointage et sélection visuelle du caractère suivant) se chevauchent. Une variante de cette hypothèse est que le pointage soit initié avant la fin de la sélection visuelle et la trajectoire corrigée en fonction de celle-ci.

Une seconde hypothèse est que certains enchaînements de frappes, notamment de touches voisines les unes des autres, sont effectués spontanément sans nécessité d'une tâche de sélection tandis que d'autre enchaînements de touches, notamment lorsque le caractère est plus éloigné, nécessitent un réel processus de sélection visuelle de la touche.

Nos futurs travaux, consistant à comparer la trajectoire du pointeur et le mouvement du regard, viseront à étudier ces deux hypothèses. Mais de fait, si nous considérions le modèle de Das comme valide, les performances d'un utilisateur expert seraient limitées par le temps de sélection de la cible (de 680ms pour un clavier contenant 27 touches), soit une vitesse maximale de 17.65 wpm. Plusieurs expérimentations contredisent cette valeur [Zhai 00, Raynal 05d, Bi 10].

Section II - Intégration des acquis dans les modèles prédictifs pour novice

Les modèles proposés, permettent de mieux tenir compte du comportement d'un novice, en additionnant par exemple le temps de vérification du caractère saisi, et de mieux représenter l'évolution des performances au cours du temps. Ils proposent également une alternative à un usage à priori abusif de la loi de Hick-Hyman comme support à la modélisation de la recherche visuelle d'un caractère par un utilisateur novice.

Néanmoins, ils présentent à nos yeux deux lacunes. En premier lieu, ils ne permettent pas de tenir compte de l'impact cognitif de l'interaction. Une des hypothèses que nous formulons est qu'une interaction de pointage a un coût cognitif certainement inférieur à une interaction plus complexe, par exemple gestuelle, nécessitant une réflexion, un choix (le choix du geste) de la part de l'utilisateur [Shannon 48]. En second lieu, ils ne permettent pas de tenir compte des heuristiques dont l'utilisateur dispose pour améliorer sa stratégie de recherche visuelle

d'une touche. Ainsi, intuitivement un utilisateur ne sera pas « novice » de la même manière face à un clavier dont l'ordonnancement est par exemple alphabétique, comparé à un ordonnancement aléatoire [Norman 82, Zhai 08, Bi 10].

De manière à illustrer cette problématique, nous proposerons en premier lieu (section II.2 et II.3) d'appliquer les modèles théoriques sur les claviers Alpha, DashKey et MessagEase (présentés dans la section II.1) et observerons que les résultats obtenus sont contre-intuitifs. Nous proposerons en second lieu de modéliser et de prendre en compte dans les modèles prédictifs, les connaissances que l'utilisateur peut exploiter face aux nouveaux claviers de même que la complexité de l'interaction (II.4). Puis nous comparerons les résultats obtenus avec nos données expérimentales (II.5).

II.1. Alpha, DashKey et MessagEase

MessagEase [Nesbat 03] (cf. Figure 13) est destiné à être utilisé sur un support tactile. Il amalgame les caractéristiques d'un clavier ambigu et la saisie par reconnaissance de gestes simples. De même qu'un clavier ambigu, chaque touche du clavier contient un ensemble de caractères. L'ensemble de caractères – la touche – est désigné par pointage.

La sélection d'un caractère au sein de cet ensemble est discriminée par deux interactions différentes. Les caractères situés en bordure de touche sont sélectionnés par le dessin d'un trait en direction du caractère : pointage de la touche, dessin du trait, puis relâchement. Le caractère central est lui sélectionné par une simple pression et relâchement de la touche.

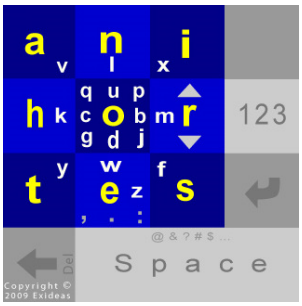


Figure 13. MessagEase

L'interaction permettant de saisir le caractère central étant plus simple et plus rapide que l'interaction destinée à saisir les caractères secondaire, MessagEase propose de disposer les caractères les plus fréquents au centre des touches.

A noter que MessagEase est initialement optimisé pour l'anglais. Pour les besoins de l'expérimentation la distribution des caractères a été optimisée pour le français.

La disposition des caractères secondaires est justifiée par l'origine de MessagEase. Initialement, le concept était destiné à un usage via un clavier de téléphone portable physique. L'interaction consistant à tracer un trait en direction du caractère était substituée par la pression successive de deux touches. Par exemple, le caractère « d » était saisi par la pression successive des caractères « 5 » puis « 8 » et à l'inverse, le caractère « w » par la pression de « 8 » puis « 5 ». Bien évidemment, dans ce contexte la saisie d'un caractère « o » (saisi par une simple pression sur 5) puis d'un caractère « e » (saisi par une simple pression sur 8) devait être désambigüisée de la saisie du « d » par l'insertion d'un délai entre la pression des deux touches.

La distribution des caractères tient donc compte de ces deux contraintes :

- Un caractère secondaire ne peut pas être situé en bordure du clavier puisqu'il n'existerait pas de seconde touche pour matérialiser le trait dans le cadre d'une version physique du dispositif;
- La répartition des caractères tend à minimiser, en fonction des caractéristiques de la langue, les ambigüités nécessitant l'insertion de délais.

Bien évidemment, dans le cas de l'usage sur un support tactile, la distribution de caractères résultante n'est plus justifiée et s'avère quelque peu perturbante.

DashKey [Merlin 10b] propose les mêmes interactions : saisie du caractère central par un simple clic et saisie des caractères secondaires par un geste en direction du caractère. Il se différencie de MessagEase par la distribution plus intuitive des caractères destinée à faciliter la mémorisation et la recherche visuelle des caractères par un utilisateur novice. Les caractères sont distribués par ordre alphabétique. Chaque touche contient 2 ou 3 caractères de telle manière que la majorité des caractères les plus fréquents puissent être positionnés au centre de la touche (cf. Figure 14).

DashKey permet ainsi de conserver au centre des touches 9 caractères représentant statistiquement 68% des caractères saisis (contre 70% pour MessagEase) tout en respectant un ordonnancement naturel par rapport aux connaissances d'un utilisateur lambda.

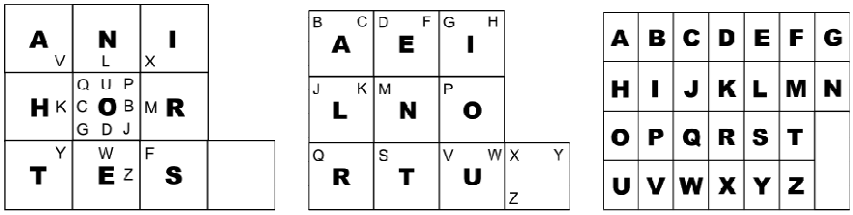


Figure 14. MessagEase, DashKey e Alpha

Le troisième clavier, Alpha, est un clavier normal dont les touches sont organisées par ordre alphabétique.

A noter que, pour les besoins de l'évaluation expérimentale, les trois claviers ont été redéveloppés. La superficie des trois claviers est identique, et les mêmes feedbacks sont utilisés pour les trois claviers.

II.2. Evaluation de Alpha, DashKey et MessagEase par les modèles prédictifs

L'évaluation à travers les modèles prédictifs impliquait en premier lieu de déterminer un modèle mécanique pour la saisie des caractères. Il n'y a bien évidemment pas d'ambiguïté particulière pour la saisie des caractères du clavier Alpha ou pour les caractères centraux des touches des claviers DashKey et MessagEase. La saisie de ces caractères correspond à de simples tâches de pointage modélisées par la loi de Fitts. Néanmoins différents coefficients (relatifs aux différents contextes d'interaction et aux artefacts support de l'interaction) ont été établis pour la loi de Fitts au terme de différentes expérimentations [Zhai 00, Zhai 02a, Zhang 98, MacKenzie 99, Accot 02], tous correspondant à cette même tâche. Trois principales paires de coefficients étant fréquemment utilisées dans diverses études comparatives, nous établirons consécutivement les résultats pour chacun d'eux.

Pour la tâche de désignation de la cible nous considérerons la complexité de la tâche W pour la loi de Fitts tel que :

$$W = \frac{key\ height + key\ width}{2}$$

En revanche, il n'existe pas de modèle rigoureusement associé à la tâche de sélection d'un caractère secondaire. Nous proposerons, en conséquence, d'évaluer cette tâche de sélection au moyen de deux modèles différents : modèle de sélection d'un item par un geste comme par exemple dans un *marking-menu* [Kurtenbach 91] ; et *discrete crossing model* proposé par Accot [Accot 02] (cf. Chapitre 1 - I.2).

Isokoski propose un clavier logiciel augmenté par *marking-menu* [Isokoski 04] permettant d'enchaîner la sélection d'un caractère par pointage puis d'un second caractère par *marking-menu* (cf. section 2.I.1). La sélection d'un caractère par *marking-menu* grève la saisie d'un temps constant de 160ms. Ce temps est extrait de l'expérience de McQueen [McQueen 95] qui évalua le temps de saisie via un *pie-menu* contenant 12 items (à noter qu'il constate une légère différence en fonction de la direction, mais il la considère lui-même comme négligeable). Par ailleurs, considérant que le nombre d'options du pie-menu sera au maximum de 8 pour MessagEase et 4 pour DashKey, la tâche est globalement plus simple que la sélection parmi 12 items proposée dans l'expérience de McQueen. En conséquence, les 160ms mesurés devraient constituer une borne supérieure par rapport au temps effectif de notre tâche de sélection ayant tendance à minimiser nos prédictions.

Une seconde possibilité, lorsque le nombre d'items dans le pie-menu est réduit (par exemple inférieur à 4 comme pour les touches de DashKey et les touches latérales de MessagEase) est de considérer les cibles individuellement. La tâche de sélection d'un caractère secondaire peut alors être assimilée à la tâche de franchissement d'une frontière (cf. Figure 15) correspondant au modèle de *discrete crossing* proposé par Accot et Zhai [Accot 02].

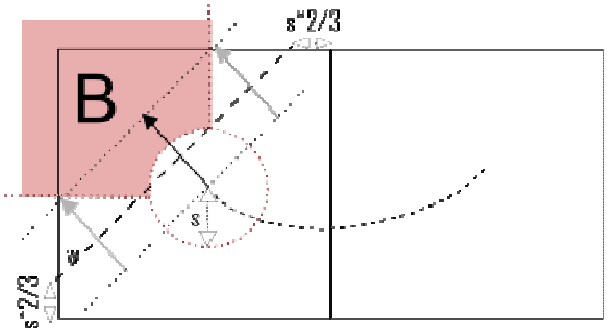


Figure 15. Sélection d'un caractère secondaire par ***discrete crossing model***

Pour la tâche de *discrete crossing*, nous considérerons la complexité de la tâche W pour la loi de Fitts par :

$$\boldsymbol{W} = \sqrt{(\boldsymbol{height} - 2\boldsymbol{s}/3)^2 + (\boldsymbol{width} - 2\boldsymbol{s}/3)^2}$$

avec *s* la distance seuil à franchir pour que la saisie soit considérée comme un trait et non pas comme un pointage simple.

Le Tableau 3 résume les différents coefficients possibles pour la loi de Fitts en fonction des différentes valeurs proposées par Zhai et al. [Zhai 00], Accot et Zhai [Accot 02], Soukoreff et MacKenzie [Soukoreff 95], et en fonction du modèle de sélection utilisé. (Pe=160ms représente la constante de saisie dans un Pie-Menu extraite de l'expérience de McQueen)

	Simple pression		Sélection par trait	
	a	b	a	b
MacKenzie	0	0.204	Pe	0.204
Isokoski/Zhai	0.083	0.127	0.083 + Pe	0.127
Accot/Zhai	0.145	0.146	0.155	0.165

Tableau 3. Paramètres utilisés dans la loi de Fitts [MacKenzie 91] pour modéliser la sélection des caractères

II.3. Modélisation, simulation et évaluation théorique

A partir de ces trois jeux de coefficients et pour les trois claviers, nous avons en premier lieu appliqué le modèle de Soukoreff et MacKenzie (cf. I.2.1) de manière à obtenir une première estimation théorique des performances pour des utilisateurs expert (cf. Tableau 4) et novice (cf. Tableau 5). A noter, le modèle a été adapté pour utilisé l'une des des paires des jeux de coéficients en fonction de la nature du caractères à sélectionner.

Experts	Alpha	DashKey	MessagEase
MacKenzie	29,7 wpm	37,9 wpm	39,7 wpm
Isokoski/Zhai	35,3 wpm	40,4 wpm	42,5 wpm
Accot/Zhai	26 wpm	34 wpm	35 wpm

Tableau 4. Modèle de Soukoreff et MacKenzie : résultats pour un utilisateur expert

Quelque soit les modèles considérés, la tâche de saisie, facilitée par une plus grande taille des touches pour les claviers DashKey et MessagEase, devrait permettre à un utilisateur expert d'obtenir de meilleurs résultats avec ces deux claviers qu'avec le clavier Alpha. Par ailleurs, comme 70% des caractères sont accessibles par un simple clic avec MessagEase contre 68% avec DashKey, ce premier devrait s'avérer sensiblement plus efficace que le second.

Le calcul des performances d'un utilisateur novice, consiste, dans le modèle de Soukoreff et MacKenzie, à additionner le temps de recherche d'une touche (obtenu par la loi de Hick-Hyman) au temps de saisie par un utilisateur expert. Dans la mesure où tous les claviers proposent le même nombre de caractères (les 26 caractères de l'alphabet latin plus l'espace), tous les temps de saisies sont grevés de la même constante Tr tel que :

$$\boldsymbol{TR} = \frac{1}{4{,}9} \times \log_2(27) = 0{,}951\text{sec}$$

De la même manière, si l'on considère désormais les modèles pour un utilisateur novice de Das [Das 08] et Pavlovych [Pavlovych 04], tous les claviers sont cette fois-ci grevés de la même constante (Tr = 680 ms) mesurée expérimentalement.

En conséquence, si les performances sont naturellement impactées par l'addition du temps de recherche du caractère, l'**ordre** entre les performances des différents claviers reste lui inchangé (cf. Tableau 5).

Novices	Alpha	DashKey	MessagEase
MacKenzie	7.8 wpm	8.6 wpm	8.7 wpm
Isokoski/Zhai	8.3 wpm	8.8 wpm	8.9 wpm
Accot/Zhai	7.7 wpm	8.5 wpm	8.6 wpm

Tableau 5. Modèle de Soukoreff et MacKenzie : résultats pour un utilisateur novice

Ces résultats théoriques prédits pour les novices sont assez contre intuitifs, notamment dans la mesure où l'ordonnancement alphabétique est une heuristique assez naturelle qui devrait orienter la recherche d'un caractère. Ainsi, un utilisateur novice devrait être initialement moins décontenancé par les distributions de touches des claviers DashKey et Alpha que par la distribution de touches de MessagEase. Par ailleurs, l'interaction avec les claviers MessagEase et DashKey étant plus complexe, il est fort probable que cette complexité impacte également les performances au cours des premiers usages.

En conséquence, on pourrait intuitivement envisager un ordre inversé par rapport à ceux prédits pour les performances d'un utilisateur novice : Alpha devrait être plus performant que DashKey, qui serait lui-même plus performant que MessagEase.

II.4. Intégration des connaissances de l'utilisateur dans les modèles théoriques

Lorsque, dans les modèles de Soukoreff et MacKenzie ou de Das, nous additionnons au temps mécanique de pointage la constante relative à la recherche d'un caractère, nous considérons les 27 caractères comme équiprobables aux yeux de l'utilisateur. Pourtant, dans de nombreuses circonstances et notamment lorsque la distribution de touches repose sur une organisation culturellement répandue telle que l'ordre alphabétique, l'utilisateur ne va pas rechercher une touche de manière aléatoire sur l'ensemble du clavier mais adresser directement sa recherche dans un sous-ensemble de caractères.

Nous proposerons donc en premier lieu de corriger le temps de recherche d'une cible en déterminant ces sous-ensembles. Nous considèrerons deux heuristiques prédominantes dans la recherche d'un caractère :

- Pour Alpha et DashKey l'utilisateur sera spontanément orienté par l'ordre alphabétique ;
- MessagEase, lui, présente deux groupes distincts de caractères : les caractères les plus fréquents situés au centre des touches et les caractères secondaires situés en bordure des touches. L'utilisateur pourra exploiter sa perception de la fréquence du caractère pour le rechercher préférentiellement dans l'un ou l'autre des deux sous-groupes.

Deux expérimentations préliminaires nous ont permis de caractériser la perception de l'ordre alphabétique et la perception de la fréquence des caractères. Nous décrirons ces deux expérimentations (a et b) puis comment nous avons intégré leurs résultats dans les modèles prédictifs (c). En second lieu, nous proposerons d'intégrer la complexité de l'interaction dans le calcul (d) et proposerons un modèle final tenant compte des ajustements proposés par Pavlovych (e).

II.4.a. Expérimentation : perception de l'ordre alphabétique

Cette première expérimentation avait pour objectif de déterminer dans quelle zone un utilisateur va spontanément rechercher un caractère en considérant que ceux-ci sont ordonnancés par ordre alphabétique.

L'expérimentation a engagé 20 utilisateurs durant une session d'environ 10 minutes. Les utilisateurs étaient âgés de 24 à 52 ans, de langue maternelle française, et tous issus du même milieu socioprofessionnel (ingénieurs, doctorants et chercheurs en informatique).

Au cours de cette session, 260 caractères (10 occurrences de chacun des 26 caractères de l'alphabet latin) étaient présentés à l'utilisateur dans un ordre aléatoire. L'utilisateur devait pointer, à l'aide d'une souris, la position supposée du caractère proposé dans un « clavier » logiciel dont seul le contour extérieur était dessiné (ni les contours des touches, ni le caractère de la touche n'étaient visibles). Les touches étaient organisés sous la forme d'une ligne unique. L'utilisateur était préalablement informé de cet agencement invisible des caractères.

L'utilisateur devait effectuer la tâche le plus rapidement et le plus spontanément possible. Les saisies supérieures à 2 secondes et demie ont été ignorées. Les erreurs n'étaient pas notifiées à l'utilisateur. La distance entre le caractère cible et la touche effectivement pointée était en revanche mesurée.

La moyenne des distances entre le caractère présenté et le caractère pointé délimite, pour chaque caractère, une zone d'incertitude dans laquelle l'utilisateur va effectuer sa recherche visuelle du caractère. La zone d'incertitude $Z(C_i)$ pour chaque caractère C_i est calculée par la relation :

$$\boldsymbol{Z(C_i)} = \frac{\sum_{\boldsymbol{k}=0}^{10} \max(\frac{|\boldsymbol{X_i} - \boldsymbol{X_k}|}{\boldsymbol{W}} - \frac{\boldsymbol{W}}{2}, 0)}{10}$$

où X_i est la coordonnnée sur l'axe horizontal du centre de la touche contenant le caractère C_i, X_k la coordonnée sur l'axe horizontale de chacune des 10 frappes de l'utilisateur tentant de cibler le caractère C_i, et W la largeur des touches.

La Figure 16 présente les résultats de l'expérimentation. Les caractères situés au début et à la fin de l'alphabet sont naturellement plus facilement situés (zone d'incertitude entre 0 et 2 touches pour les caractères A, B, C, D, X, Y, Z). Le milieu de l'alphabet semble être un deuxième critère de référence pour localiser les caractères (distance entre 2 et 2.5 pour les caractères M, N et O). Enfin, plus les caractères s'éloignent de ces trois points de références, plus l'imprécision est grandissante.

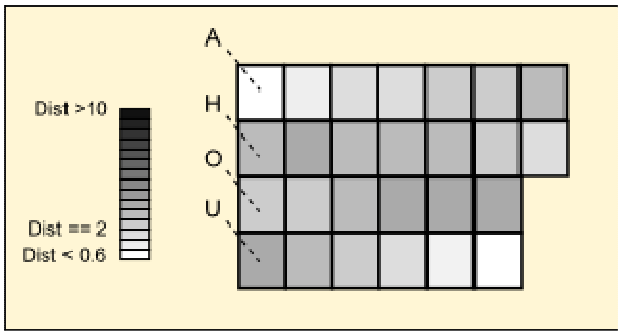


Figure 16. Perception de la position d'un caractère dans un clavier alphabétique

II.4.b. Expérimentation : perception de la fréquence des caractères

La seconde expérimentation consiste à déterminer dans quelle mesure un utilisateur considère spontanément qu'un caractère fait partie des 9 caractères les plus fréquents présents dans la langue française et ainsi s'il recherchera préférentiellement tel ou tel caractère parmi les caractères centraux ou secondaires de MessagEase.

L'expérimentation a engagé également 20 utilisateurs dans une session d'environ 10 minutes. De même que dans l'expérimentation précédente, 260 caractères (10 occurrences de chacun des 26 caractères de l'alphabet latin) étaient présentés dans un ordre aléatoire. Deux boutons permettaient à l'utilisateur de classer le caractère présenté parmi les 9 caractères les plus fréquents ou parmi les 17 caractères les moins fréquents de l'alphabet. Les erreurs n'étaient pas signifiées à l'utilisateur.

Les résultats (cf. Figure 17) ci-dessous illustrent que les utilisateurs ont classé sans ambiguïté 6 caractères parmi les plus fréquents et 11 caractères parmi les moins fréquents. Les 9 autres caractères sont en revanche fréquemment classé dans le mauvais groupe.

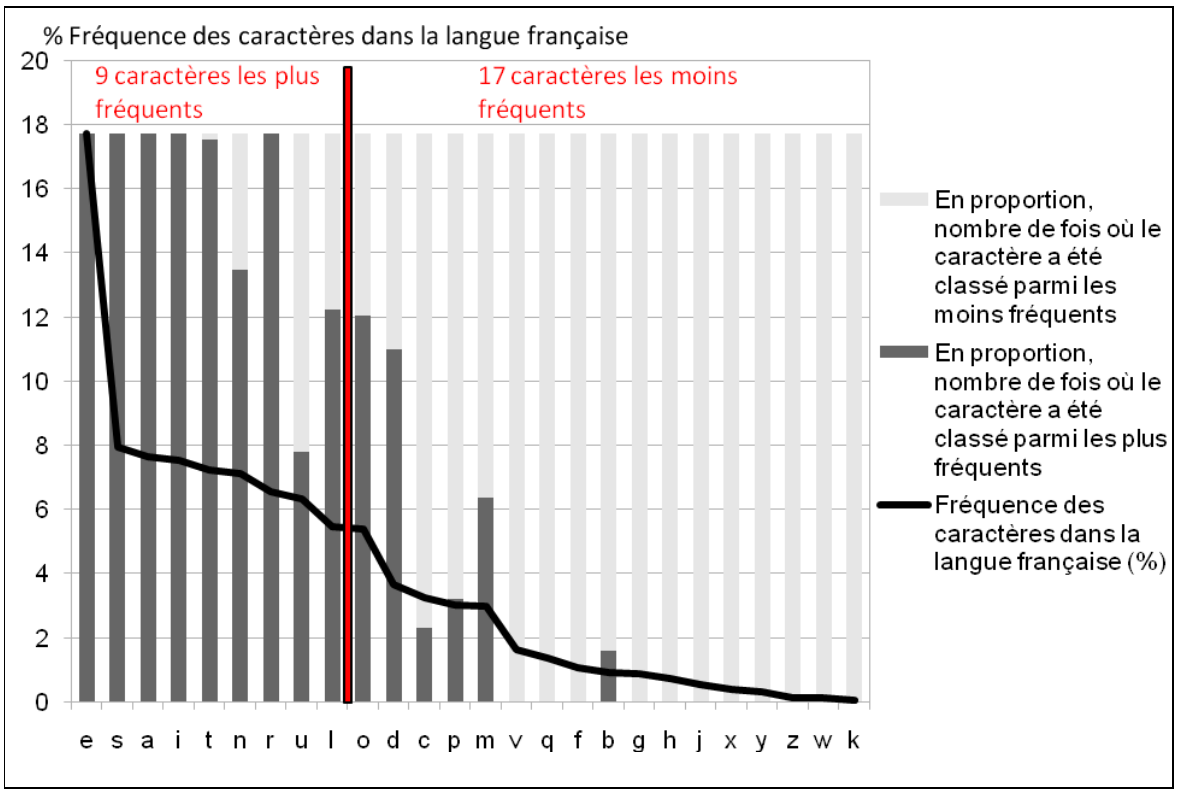


Figure 17. Perception de la fréquence des caractères dans la langue française

II.4.c. Intégration des données heuristiques dans les modèles théoriques

Pour les trois claviers, nous considérerons que la position du caractère « espace » est immédiatement apprise en raison de sa position particulière et de son apparence distincte, de même qu'en raison de sa haute fréquence de saisie.

En premier lieu, pour le clavier Alpha et DashKey, au lieu de considérer un espace de recherche du caractère de 27 caractères, pour chaque caractère φ, nous considérons un espace correspondant à la zone d'incertitude Z (φ) mesurée durant les expérimentations préliminaires. En conséquence, le temps de recherche est différent pour chaque caractère et est établi par la formule :

$$TR(\ ) = \frac{Z(\ ) \times 680}{27}$$

où 680 ms est le temps de recherche d'un caractère parmi 27 établis par [Pavlovych 04].[11]

Les caractères centraux et secondaires de MessagEase sont visuellement nettement distincts. On considère que l'utilisateur va donc rechercher indépendamment dans l'un ou l'autre des deux groupes de caractères. Lorsqu'il estime que le caractère est fréquent, il va rechercher ce caractère parmi les 9 caractères centraux et à contrario parmi les 17 caractères secondaires lorsque celui-ci est perçu comme moins fréquent. Cependant, 9 caractères sont fréquemment classifiés dans le mauvais groupe ce qui implique en pratique que, dans un certain pourcentage de cas, l'utilisateur va en premier lieu chercher à localiser le caractère dans un groupe puis, dans un second temps, le rechercher parmi les caractères du deuxième groupe.

En conséquence, nous avons trois temps de recherches du caractère distincts en fonction des trois situations :

- Pour les 6 caractères E, T, A, R, I, S classifiés sans ambigüité parmi les plus fréquents, le temps de recherche du caractère parmi les 9 caractéres centraux est :

$$TR(\ ) = \frac{9 \times 680}{27}$$

- Pour les 11 caractères K, W, Z, Y, X, J, H, G, F, Q, V classifiés sans ambigüité parmi les moins fréquents, le temps de recherche du caractère parmi les 17 caractères secondaire est :

$$TR(\ ) = \frac{17 \times 680}{27}$$

- Pour les 9 autres caractères, nous considérons E (φ) la probabilité, issue de la seconde expérimentation préliminaire, de rechercher le caractère φ dans le mauvais groupe. L'utilisateur devra nécessairement rechercher le caractère dans le bon groupe mais, avec une probabilité de E (φ), il va en premier lieu rechercher le caractère dans le mauvais groupe soit :
    - pour les caractères fréquents N, U et L un temps de recherche :

$$TR(\ ) = \frac{680}{27} \times (9 + E(\ ) \times 17)$$

    - pour les caractères les moins fréquents O, D, C, P, M et B un temps de recherche :

$$TR(\ ) = \frac{680}{27} \times (17 + E(\ ) \times 9)$$

II.4.d. Impact cognitif de l'interaction

Lorsque l'interaction est une simple tâche de pointage, celle-ci étant opérée mécaniquement, on ignore un quelconque travail cognitif relatif à la tâche. Cependant, dans les cas des claviers MessagEase et DashKey, l'interaction est plus complexe et moins spontanée notamment pour un utilisateur novice. Celui-ci doit choisir entre une simple interaction de

[11] Dans le mesure où la tâche est linéaire, le temps de recherche pour chacun des 27 caractères est de 680/27. La zone d'incertitude contenant Z(φ) caractères, le temps TR(φ) est donc proportionnel à 680/27 en fonction de ce nombre de caractères soit TR(φ)= Z(φ)*680/27.

pointage et le tracé d'un trait. Un travail cognitif est imputable à un choix entre deux alternatives. Ce choix peut être modélisé par la loi de Hick-Hyman (cf. Chapitre 1 - I.3).

Pour les claviers DashKey et MessagEase, on considère donc un temps supplémentaire TC pour la réalisation de la tâche tel que : [12]

$$\boldsymbol{TC} = \frac{1}{4{,}9} \times \log_2(2)$$

II.4.e. Corrections relatives au modèle de Pavlovych

Pavlovych observe qu'il est difficile de comparer le « lower-bound » avec des résultats expérimentaux dans la mesure où il modélise uniquement la tâche de pointage sans tenir compte de la tâche de recopie. Il propose en conséquence d'ajouter deux temps au temps de recherche et de pointage d'un caractère : le temps de lecture du mot (TL) et le temps de vérification de la frappe (TV).

La tâche (saisie de mot isolé) et le mode de calcul du temps de saisie (temps de saisie mesuré entre la frappe du premier caractère et la frappe du dernier caractère du mot) que nous utiliserons dans l'expérimentation nous permettent d'exclure le temps de lecture du mot. En revanche, nous considérerons dans le modèle de saisie le temps de vérification de la frappe TV=0.14s mesuré par Pavlovych.

II.4.f. Résultats

La saisie est donc divisée en quatre sous-tâches: la recherche de la touche dont le temps TR (φ) est dépendant du caractère et du clavier considéré, le choix de l'interaction dont le temps TC est dépendant du clavier, l'interaction dont le temps T (φ) est dépendant du modèle de sélection du caractère (pression simple versus geste), et enfin la vérification du caractère saisi dont le temps est constant TV.

Le Tableau 6 résume les résultats, en fonction des différents modèles préalablement considérés pour la sélection des caractères, et en tenant des nouveaux modèles pour la représentation de la saisie.

Novices	MessagEase	DashKey	Alpha
MacKenzie	8.66 wpm	10.15 wpm	11.30 wpm
Isokoski/Zhai	8.77 wpm	10.39 wpm	12.04 wpm
Accot/Zhai	8.63 wpm	10.04 wpm	10.98 wpm

Tableau 6. Résultats obtenus avec les nouveaux modèles pour un utilisateur novice

Quelque soit les valeurs du tableau 3 utilisées, l'ordre proposé entre les trois claviers pour les performances d'un novice, en intégrant dans les modèles l'héritage culturel de l'utilisateur et les coûts cognitifs relatifs à la complexité de l'interaction, apparaît plus en adéquation avec l'ordre intuitivement établi. Reste à confronter ces résultats avec des valeurs expérimentales

[12] Dans le cas du clavier Alpha, vu qu'il n'y a pas d'alternative, il s'agit d'un choix ayant une possibilité dont le coût est TC = 1/4,9 * log2 (1) = 0

II.5. Évaluation expérimentale

Nous avons, en conséquence, effectué une expérimentation destinée à évaluer, pour les trois claviers, les performances d'un utilisateur au cours des premiers usages.

II.5.a. a) Dispositifs et participants

L'expérimentation a engagé 18 participants volontaires (11 hommes et 7 femmes), tous droitiers et usagers quotidiens du clavier AZERTY. Les utilisateurs étaient âgés de 24 à 36 ans, de langue maternelle française, et tous issus du même milieu socioprofessionnel (ingénieurs, doctorants et chercheurs en informatique).

Le support de l'expérimentation était un téléphone portable HTC Touch HD équipé d'un écran tactile et fonctionnant sous Windows Mobile 6.1. Les trois claviers destinés à l'expérimentation ont été développés en C++. Les utilisateurs interagissaient sur l'écran tactile du téléphone avec les claviers logiciels par l'intermédiaire d'un stylet.

II.5.b. b) Procédures

Chacun des 18 utilisateurs était conduit à effectuer une tâche de recopie de mots avec chacun des claviers DashKey, MessagEase et Alpha. Au cours d'une session, l'utilisateur effectuait la saisie de 42 mots avec l'un des trois artefacts. Les trois sessions étaient espacées de 2 minutes. Les 42 mots étaient identiques pour les trois sessions mais présentés dans un ordre aléatoire. Le corpus de mots a été choisi de manière à respecter les caractéristiques de la langue française en matière de fréquence des lettres et fréquence des digraphes.

Le mot à recopier et les caractères saisis par l'utilisateur étaient représentés dans deux bandeaux superposés (cf. Figure 18). Les erreurs n'étaient ni affichées ni corrigées mais la saisie du caractère correct était exigée. Un feedback visuel et audio alertait l'utilisateur de l'erreur.

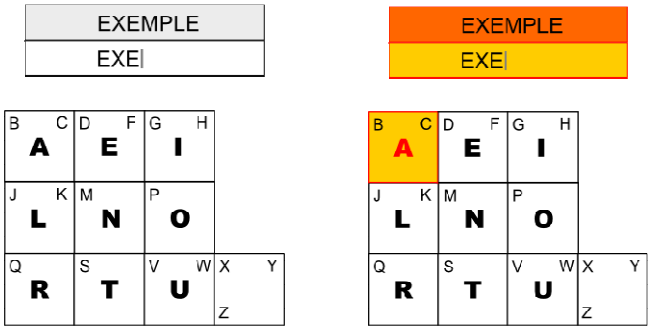


Figure 18. Dispositif expérimental (gauche) et feedback d'erreur (droite)

Avant chaque session, l'utilisateur était informé des caractéristiques du clavier utilisé au cours de la session à venir : technique d'interaction et principes d'organisation de la distribution de touches. Les 18 utilisateurs ont été divisés en 6 groupes de manière à alterner l'ordre d'utilisation des trois artefacts et contrebalancer d'éventuels effets d'apprentissage (notamment sur la technique d'interaction partagée entre MessagEase et DashKey) ou de fatigue.

L'expérimentation était précédée d'une brève session de découverte du principe d'interaction proposé par MessagEase et DashKey. Au cours de cette session de découverte, l'utilisateur devait effectuer la saisie de 10 caractères via un clavier réduit à deux touches (cf. Figure 19). Ce « clavier » proposait une technique d'interaction similaire à celle de MessagEase et DashKey, mais une distribution de touches complètement différente de manière à éviter un apprentissage de l'un ou l'autre des deux distributions de touches.

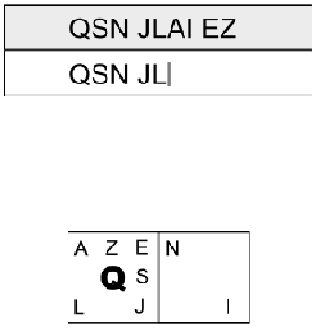


Figure 19. Clavier d'entraînement à la technique d'interaction

L'expérimentation était complétée d'un bref questionnaire recueillant l'opinion de l'utilisateur sur la technique d'interaction proposée, sur la distribution de touches et les perspectives d'utilisation des claviers dans un usage réel.

II.5.c. c) Résultats

Les résultats statistiques illustrent que l'ordre des exercices, contrebalancé par la répartition des utilisateurs en 6 groupes, n'a pas eu de conséquences sur les résultats finaux.

Les résultats de l'expérimentation (cf. Tableau 7) confirment ainsi les hypothèses intuitives et les modèles proposés par rapport au classement des performances d'un utilisateur débutant dans l'usage des trois claviers. En termes de vitesse de saisie (et également en termes de nombre d'erreurs), nos utilisateurs ont été plus efficaces avec le clavier Alpha, puis avec le clavier DashKey et enfin avec MessagEase.

Novices	MessagEase	DashKey	Alpha
MacKenzie	8.66 wpm	10.15 wpm	11.30 wpm
Isokoski/Zhai	8.77 wpm	10.39 wpm	12.04 wpm
Accot/Zhai	8.63 wpm	10.04 wpm	10.98 wpm
	WPM / error (%)	WPM / error (%)	WPM / error (%)
Experimental	9.13 / 2.9	11.1 / 2.17	11.6 / 0.73

Tableau 7. Comparaison entre modèle et résultats expérimentaux

Les valeurs expérimentales obtenues sont proches des valeurs calculées au moyen des différents modèles bien que les modèles proposés pour DashKey et MessagEase restent légèrement pessimistes.

D'un point de vue qualitatif, les utilisateurs ont par ailleurs préféré unanimement l'organisation des caractères proposée par DashKey par rapport à l'organisation des caractères de MessagEase. Cette première organisation a été jugée plus intuitive, l'organisation alphabétique facilitant concrètement la recherche d'un caractère. Ces observations corroborent naturellement les résultats numériques.

En ce qui concerne la technique de sélection des caractères de DashKey et MessagEase, les utilisateurs l'ont qualifiée de récréative, voire ludique. Néanmoins, elle est également considérée comme plus exigeante en termes de concentration et donc d'utilisation plus

fatigante et générative d'un grand nombre d'erreurs. Bien qu'ils estiment avoir une nette marge de progression avec ces deux claviers, aucun des utilisateurs ayant participé à l'expérimentation n'envisagerait concrètement leur usage à long terme pour de la saisie effective de textes.

Section III - Synthèse

III.1. Novice versus débutant

Outre la problématique engendrée par l'utilisation de la loi de Hick-Hyman, la notion de novice modélisée par le lower-bound [Soukoreff 95] serait complètement valide si l'utilisateur explorait pour la première fois une nouvelle disposition des caractères, ceci sans qu'une heuristique particulière ne puisse le guider dans la recherche d'un caractère particulier (par exemple pour comparer des claviers du type GAG [Raynal 05b], ATOMIK [Zhai 02a] ou encore Metropolis [Zhai 00]). Cependant, dans la mesure où l'utilisateur est en mesure d'apprendre très rapidement certaines caractéristiques du clavier [Pavlovych 04, Zhai 02b], le modèle reste valide pendant une durée réduite et indéterminée de temps.

En outre, le modèle n'intègre pas d'autres coûts cognitifs que la recherche des touches dans un espace statique où sont distribués les caractères. Ainsi, non seulement le modèle ne permet pas de tenir compte de modifications dynamiques du clavier (comme par exemple l'utilisation des listes de prédiction), mais il ne permet pas non plus de rendre complètement compte de la tâche de saisie [Pavlovych 04]. En réponse à ce deuxième point, les modèles proposés par Pavlovych et Das [Das 08] visent à corriger ce modèle en tenant mieux compte du comportement réel d'un utilisateur novice face au clavier.

Néanmoins, ces modèles corrigés considèrent toujours l'utilisateur novice comme dépourvu de connaissances susceptibles de l'orienter dans son usage du clavier. Dans de nombreuses situations (et notamment lorsque le clavier est supporté par une distribution de touches connue tel que l'AZERTY ou l'ordre alphabétique), cette notion de novice ne correspond pas à une réalité tangible. Cet utilisateur virtuel « novice » est beaucoup plus ingénu qu'un véritable utilisateur « débutant » avec le clavier.

Les modèles proposés restent donc génériques mais ils ne permettent pas de rendre compte de façon satisfaisante, et surtout représentative de la réalité, de la comparaison des performances via deux claviers dont les propriétés de l'un et/ou l'autre sont partiellement connues. C'est notamment le cas lorsque nous comparons les claviers Alpha, MessagEase et DashKey. Confronté à Alpha, l'utilisateur est en quelque sorte novice dans la mesure où il n'a jamais été amené à utiliser un tel clavier dans son quotidien. Pourtant, il n'est pas désorienté de la même manière que si on lui propose une distribution de touches quelconque. Il est de même beaucoup plus familier avec la technique d'interaction de Alpha qu'avec la technique d'interaction proposée par les deux autres claviers. L'héritage « culturel », l'expérience de l'utilisateur impact donc sa condition de novice.

En conséquence, nous avons proposé d'intégrer l'expérience culturelle de l'utilisateur dans les modèles pour ne considérer non plus un utilisateur « novice », mais un utilisateur « débutant ». Notre démarche fût, en premier lieu, d'intégrer l'expérience d'un utilisateur en mesure d'influencer la recherche visuelle des caractères. Un utilisateur, culturellement familier avec la notion d'ordre alphabétique, ne va pas avoir la même stratégie pour rechercher un caractère au sein des claviers Alpha et DashKey ou MessagEase. La prise en compte de ce facteur nous a permis de corriger la prévision des performances et de rendre compte que MessagEase sera nécessairement, relativement à notre population d'utilisateurs,

le moins performant vis à vis d'un utilisateur débutant, exposé pour la première fois aux trois claviers.

En second lieu, nous avons intégré l'expérience relative à l'interaction et matérialisé le fait qu'un utilisateur débutant n'agira pas de façon aussi spontanée face à une technique d'interaction simple et complètement maîtrisée, telle de simples pressions de touches, qu'avec la technique d'interaction proposée par DashKey et MessagEase. La prise en compte de ce deuxième facteur nous a permis de nous rendre compte que Alpha sera nécessairement plus performant que DashKey (respectivement que MessagEase) pour un utilisateur débutant, bien que les deux claviers soient agencés sur la base de l'ordre alphabétique.

Cependant, la prise en compte de ces deux aspects, aspects tous deux relatifs à l'expérience de l'utilisateur, font émerger que le modèle d'évaluation de débutant ne peut plus être universel. Ainsi, bien que nous n'en ayons pas fait l'expérience concrète, la distribution alphabétique n'aura probablement pas une conséquence également significative auprès d'une population latine qu'auprès d'une population utilisant l'alphabet cyrillique. De même, une population de personnes âgées – sans contacts réguliers avec l'informatique – sera sans-doute plus à son aise face à une distribution alphabétique que face à une distribution AZERTY. Ceci ne sera probablement pas le cas pour une population d'utilisateurs aguerrie à l'usage de claviers virtuels.

En conséquence, les modèles que nous proposons et la notion même de débutant, sont nécessairement relatifs à une classe de population. Ainsi, un modèle « débutant » n'est pas la modélisation d'une caractéristique intrinsèque et absolue du clavier, mais un modèle relatif à l'interaction d'une classe d'utilisateur avec le clavier.

III.2. Conséquence sur la comparaison de différents claviers

De même que Pavlovych, Das, Kristensson [Kristensson 07] ou encore Bi [Bi 10], nous considérons que les performances des utilisateurs débutants sont une donnée importante déterminant les perspectives d'utilisation du clavier dans un contexte non contrôlé. La prise en compte de cette caractéristique est donc essentielle dans l'évaluation d'un clavier.

L'évaluation d'un clavier doit non seulement être considérée relativement aux propriétés motrices des utilisateurs, mais elle doit également être sensible aux caractéristiques socioculturelles de la population cible.

III.3. Pertinence de l'utilisation des modèles prédictifs dans l'évaluation des performances d'un utilisateur débutant

Dans le cadre de nos travaux visant à améliorer la représentativité des modèles prédictifs pour les performances d'un utilisateur débutant, nous avons ainsi adopté des stratégies contextuelles permettant de prendre en compte ces aspects socioculturels des utilisateurs cibles de l'expérimentation. Néanmoins, ces stratégies exigeraient nécessairement que le modèle soit adapté pour chacune des populations cibles et pour chacun des claviers. Pour ce faire, il fût par ailleurs nécessaire de réaliser plusieurs expérimentations préliminaires permettant de mesurer les paramètres relatifs aux données socioculturelles requis pour chaque modèle.

Mais nous devons surtout admettre que la conception du modèle pour chacun des claviers a été notamment guidée par une forte intuition quant aux résultats qui devaient être obtenus. Nos travaux ne nous ont pas permis d'identifier une démarche systématique permettant d'intégrer ces données dans les modèles sinon: d'être attentif à l'expérience de l'utilisateur par rapport à la distribution de touches proposée et d'être attentif à la complexité de

l'interaction. En conséquence, nous disposons de modèles prédictifs qui peuvent s'avérer pertinents pour comparer les performances d'utilisateurs débutants, mais ils doivent être confirmés expérimentalement.

La mise au point de cette donnée prédictive requiert au final une démarche expérimentale beaucoup plus lourde que la simple évaluation expérimentale du clavier. Ainsi, si les modèles nous permettent de mieux décrire le comportement d'un utilisateur novice face à un clavier, cette exigence nuit considérablement à la perspective de les utiliser efficacement comme réel outil d'évaluation et de comparaison.

Ces travaux ont néanmoins montré que, si le moyen le plus efficace de fournir une donnée numérique sur les performances réelles d'un utilisateur débutant restait l'expérimentation, une simple comparaison heuristique des claviers permettrait tout de même de fournir des données pertinentes concernant l'ordre probable des performances entre les claviers et concernant les difficultés que l'utilisateur est susceptible de rencontrer par rapport à un clavier, voire concernant les perspectives d'utilisation à court terme (et donc à long terme) des claviers.

Chapitre 4 - Évaluation heuristique des claviers logiciels

Du fait de la nature de la tâche générant une succession continue de pointages, la saisie de texte par l'intermédiaire de claviers logiciels fût, dans un premier temps, un support pertinent d'études pour les lois prédictives telles que Fitts et Hick-Hyman ou KLM. La tâche simple et répétée, engendrant des coûts cognitifs *a priori* maîtrisés, offrait l'un des deux cas d'application concrets pour ces lois dans le domaine de l'IHM de même que l'étude des menus [Kurtenbach 91, Ahlström 05, Accot 97, Cockburn 07].

Dans un second temps, ces lois prédictives sont devenues réciproquement un moyen de fournir des éléments de comparaison pour l'évaluation des claviers logiciels [Soukoreff 95] sans néanmoins que ne soient discutées les limites d'application, la représentativité concrète, ni la finalité des valeurs fournies. Plus généralement, les évaluations théoriques ou expérimentales des claviers logiciels ont visé, le plus souvent, à faire état d'une compétitivité de l'artefact étudié sur la base d'une donnée considérée comme objective: les performances d'un utilisateur expert.

Comme nous avons pu l'observer, non seulement cette donnée n'est plus objective lorsque nous venons à comparer les nouvelles solutions de claviers complexes, mais, de plus, elle ne serait représentative que de l'une des facettes du clavier ne permettant de qualifier ni l'utilisabilité d'un clavier au quotidien, ni *a fortiori* les perspectives d'appropriation du clavier par une population.

En conséquence, dans ce chapitre, nous proposons de discuter de la finalité des évaluations (I) ainsi que des autres métriques nécessaires pour caractériser les performances d'un clavier par rapport à une population cible et un contexte d'usage (II). Nous proposerons enfin quelques comparaisons (III).

Section I - Finalité des évaluations

La critique générale que nous avons formulée jusqu'à présent, concernant les évaluations des claviers logiciels, est que celles-ci comparent les claviers sur la base d'une donnée numérique, les performances d'un utilisateur expert, sans comparer les conditions requises pour obtenir et maintenir ces performances, et sans même définir si un groupe d'utilisateurs cible est en mesure d'atteindre ces performances dans le cadre d'un usage normal.

Cette base de comparaison met donc en totale parenthèse la finalité des claviers logiciels : permettre à une population cible (utilisateurs lambdas, adolescents, personnes âgées, personnes handicapées, etc.), dans un contexte éventuellement spécifique (contexte de mobilité, contraintes particulières relatives au dispositif physique d'entrée, handicap moteur, usage ponctuel ou fréquent, etc.), d'effectuer de la saisie de textes brefs, longs, en langage 'SMS' [Grinter 01, Grinter 03], formatés ou non.

Nous souhaitons donc défendre que les évaluations devraient d'une part permettre de mesurer l'adéquation d'un clavier à un ou plusieurs objectif(s) concret(s), mais également permettre de mesurer les perspectives et conditions d'adoption de l'artefact par une population. Ainsi, les performances ne doivent pas se caractériser par une valeur unique pour un utilisateur singulier – un utilisateur expert –, mais par un gain moyen sur une

population par rapport à un dispositif de référence usuellement employé dans le(s) contexte(s) d'étude envisagé(s).

Néanmoins, cette façon d'aborder l'évaluation mènerait à appréhender le clavier dans un contexte d'usage réel. Cela se traduirait expérimentalement par une étude sur un échantillon relativement important de la population cible et sur une période relativement étendue. Ces contraintes sont bien évidemment peu compatibles avec les possibilités matérielles et économiques. Il est donc pertinent d'identifier des critères ergonomiques, en réponse aux caractéristiques du contexte et de la population cible, afin d'effectuer une évaluation heuristique [Nielsen 90, Nielsen 93, Nielsen 94] des différents claviers.

Section II - Métriques nécessaires à la caractérisation des claviers logiciels

Les propriétés des différents claviers doivent donc être mises en confrontation par rapport à un but – la nature du texte à saisir – atteint par une population cible, dans un contexte d'usage donné. Nous nous proposons donc ici d'identifier différents profils de but, de populations et de contextes et d'opposer les aspects des claviers susceptibles de favoriser ou pénaliser ces aspects[13].

II.1. Les buts

Sur la base d'un questionnaire effectué auprès de 131 personnes (pour 2/5 françaises et 3/5 brésiliennes) par la voie d'un site web[14] (108 questionnaires) et par voie directe (23 questionnaires), nous avons proposé une classification des usages des claviers logiciels. Les claviers logiciels peuvent être dédiés à différents usages allant de la saisie de phrases simples dans un langage presque phonétique [Grinter 01] (type SMS) jusqu'à des documents formels longs et formatés. Nous proposons de détailler les catégories suivantes :

- SMS informel (nous qualifions ici l'usage codifié, pseudo-phonétique, des SMS) : le texte est court, peu formaté et contient une ponctuation minimaliste ; le texte ne respecte pas les propriétés syntaxiques, grammaticales et orthographiques de la langue. Les caractères accentués sont ignorés.

- SMS formel (nous qualifions ici un usage linguistiquement correct des SMS) : le texte est court, peu formaté et contient une ponctuation minimaliste ; le texte respecte globalement les propriétés syntaxiques, grammaticales et orthographiques de la langue. Les caractères accentués sont néanmoins parfois ignorés.

- Email : le texte est de taille quelconque, peu formaté et contient une ponctuation complète ; le texte respecte les propriétés syntaxiques grammaticales et orthographiques de la langue. Le texte respecte les caractères accentués.

- Document : le texte est de taille quelconque, formaté et contient une ponctuation complète ; le texte respecte les propriétés syntaxiques grammaticales et orthographiques de la langue. Le texte respecte les caractères accentués.

Sur la base des questionnaires, nous avons pu observer que les deux premiers types d'usages représentent les usages les plus fréquents. Les utilisateurs plus jeunes (adolescents et étudiants) sont globalement les plus gros consommateurs mensuels de SMS (malgré une très grosse disparité interindividuelle) et rédigent ces SMS de façon

[13] Cf. recommandation ISO 9241-11

[14] Questionnaire accessible à l'adresse http://www.intnovate.org/sms

majoritairement informelle (~60% des cas). Les utilisateurs plus âgés rédigent très majoritairement leur SMS de façon formelle (plus de 80% des cas). La rédaction d'emails au moyen d'un clavier virtuel reste très occasionnelle. Quant à l'usage de clavier logiciel pour l'édition de documents, il est presque réservé à des utilisateurs handicapés.

Néanmoins, dans une proportion non négligeable, les objectifs de l'utilisateurs sont sensibles à des exigences supplémentaires par rapport à la simple possibilité de saisir les 26 caractères de l'alphabet latin.

La rédaction d'emails et de documents, sensible au respect des caractères linguistiques référents à la ponctuation et aux caractères accentués doit, par exemple, exiger un accès efficace à ces mêmes caractères.

Pour ces mêmes types de textes, d'éventuelles propositions de complétion peuvent accélérer la saisie et permettre même d'aider l'utilisateur à mieux respecter l'orthographe. En revanche, ces genres de dispositifs vont s'avérer inefficaces voir totalement intrusifs et contreproductifs [Grinter 01, Grinter 03] pour la saisie de SMS informels.

La rédaction d'un document formaté nécessite, par ailleurs, l'utilisation de fonctionnalités de mise en forme potentiellement accessibles, au niveau du clavier, par des combinaisons de touches.

En confrontant les claviers existants aux différents buts des utilisateurs, nous avons identifié les caractéristiques suivantes permettant d'influencer positivement ou négativement l'utilisabilité du clavier par rapport aux différents objectifs envisagés:

- Facilité d'accès aux majuscules
- Facilité d'accès aux caractères de ponctuation
- Facilité d'accès aux caractères accentués
- Facilité d'accès aux caractères spéciaux
- Saisie 'intuitive' des majuscules
- Complétion
- Outil de désambiguïsation linguistique pour les claviers ambigus
- Apprentissage du système de complétion sur la base des saisies utilisateur

II.2. Populations cibles

Les claviers sont par ailleurs destinés à différents publics. Sur la base des questionnaires, nous avons identifié les principales catégories de populations cibles aux propriétés particulières:

- Utilisateurs familiarisés avec le clavier AZERTY de l'ordinateur (aujourd'hui la majorité des utilisateurs francophones potentiels),
- Utilisateurs non familiarisés au clavier AZERTY,

- Personnes âgées : au-delà d'une faible familiarisation avec le clavier AZERTY, les personnes âgées sont plus sensibles à des touches de taille réduite, des interactions plus complexes et plus exigeantes en termes de précision, etc.

- Utilisateurs partiellement ou totalement handicapés des membres supérieurs. Ces utilisateurs ont plus de difficultés à effectuer des pointages précis. L'effort nécessaire pour déplacer le dispositif de pointage peut également s'avérer fatigant.

Des caractéristiques du design des claviers peuvent faciliter l'appréhension d'un clavier par l'un ou l'autre de ces publics. Par exemple, des personnes âgées, ayant peu été en contact avec l'ordinateur, auront plus de familiarité avec un clavier structuré sur la base de l'ordre alphabétique que sur un clavier AZERTY. Au contraire, les habitudes d'un utilisateur aguerri au clavier AZERTY seront contrariés face à un clavier alphabétique [Norman 82]. La taille des touches (et éventuellement des caractères) peut en outre avoir un rôle encore plus déterminant pour une population âgée que pour une population plus jeune. Parallèlement, la complexité de l'interaction et la sollicitation du point de vue cognitif n'auront pas les mêmes réponses face à ces différentes populations.

En outre, une population telle que les utilisateurs handicapés moteurs des membres supérieurs peut être plus sensible à la diminution de la distance parcourue par le curseur au détriment d'autres facteurs.

Par ailleurs, certains aspects des claviers (nouvelles distributions de touches, aides dynamiques par exemple) peuvent nécessiter un apprentissage pour pouvoir être exploités efficacement. La fréquence d'utilisation de l'artefact peut influer sur l'apprentissage et sur la possibilité même de cet apprentissage. De même, l'utilisation au quotidien d'autres claviers peut contrarier cet apprentissage.

En confrontant les claviers existants aux différents caractéristiques des populations cibles, nous avons identifié les caractéristiques suivantes des claviers pouvant influer l'utilisabilité par rapport à la population cible:

- Ordonnancement basé sur une distribution alphabétique des caractères

- Ordonnancement basé sur une distribution AZERTY des caractères

- Ordonnancement basé sur une distribution libre (optimisée) des caractères

- Clavier ambigu basé sur une distribution alphabétique des caractères

- Clavier ambigu basé sur une distribution libre des caractères

- Interaction par de simples pressions de touches physiques ou vrituelles

- Interactions mixtes

- Aides dynamiques additionnelles

- Réagencements dynamiques

- Taille des touches

- Taille des caractères

- Paradigme aidant à la réduction des distances parcourues par le curseur

II.3. Contexte d'usage

Enfin, un troisième aspect peut influer sur l'utilisabilité d'un clavier logiciel : le contexte d'utilisation. Par exemple, les caractéristiques requises pour faciliter la saisie ne sont pas identiques lorsque l'utilisateur se dédie à cette unique tâche ou lorsque cette tâche est effectuée parallèlement à une autre activité. Nous avons ainsi identifié différents contextes d'utilisation pouvant exiger des caractéristiques distinctes :

- Activité simple de saisie : l'artefact est potentiellement contrôlé avec les deux mains, l'activité cognitive est totalement concentrée sur la tâche de saisie.

- Usage en transports collectifs : l'artefact est manipulé à une main, l'activité cognitive est principalement concentrée sur la tâche de saisie. La précision de l'utilisateur est handicapée par le mouvement.

- Usage en contexte de marche : l'artefact est potentiellement contrôlé avec les deux mains, l'activité cognitive est partiellement occupée par l'activité de marche. La précision de l'utilisateur est handicapée par le mouvement.

- La tâche de saisie est conduite en parallèle d'une activité principale (conduite par exemple). La saisie, souvent brève, est effectuée par intermittence.

- Usage aveugle : l'utilisateur interagi avec le clavier sans un rétro contrôle visuel permanent avec celui-ci.

Ainsi, autant un outil de complétion peut potentiellement apporter une aide tangible pour un utilisateur concentré sur la tâche de saisie, autant cet outil n'est d'aucun support (voire handicapant dans la mesure où il consomme de l'espace) pour un usage aveugle. La taille des touches, la simplicité de l'interaction sont d'autres critères pouvant simplifier l'usage dans des contextes particuliers tel que la mobilité ou lorsque la tâche de saisie est perturbée par une activité parallèle mobilisant des ressources cognitives de l'usager.

Nous avons en conséquence identifié certaines caractéristiques des claviers pouvant influer sur la saisie en fonction du contexte :

- Taille des touches

- Aides dynamiques en marge du clavier (listes de complétion par exemple)

- Aides dynamiques sur le clavier (KeyGlass par exemple)

- Réagencement dynamique des caractères

- Autocorrection d'erreurs

- Réagencement statique des caractères

- Complexité de l'interaction

Section III - Évaluations heuristiques

Nous avons donc identifié une combinatoire de quatre paramètres influant sur l'utilisabilité d'un même clavier : la nature du texte à saisir, les caractéristiques de l'utilisateur, la fréquence d'utilisation et le contexte d'utilisation. Un clavier efficace pour l'une de ces

combinatoires, peu s'avérer totalement inefficient, voire inutilisable, pour une autre combinatoire.

Face à ces différents paramètres, nous avons identifié plusieurs séries de critères pour les claviers, chacun pouvant influencer plus ou moins positivement, plus ou moins négativement, ou rester neutre par rapport à l'utilisation d'un clavier.

En conséquence, nous nous proposons de noter ces critères entre -5 (très négatif) et 5 (très positif), en considérant que 0 est de conséquence neutre : pour chaque clavier évalué ; et en fonction de chaque valeur des différents paramètres. Ces valeurs nous permettront d'obtenir une note du clavier relative à chaque combinatoire de paramètres.

L'évaluation heuristique n'échappant pas aux règles des évaluations en général, il serait, en outre, nécessaire de confronter les points de plusieurs évaluateurs. Nielsen [Nielsen 90] estime qu'il faut 5 évaluateurs experts pour obtenir une évaluation heuristique représentative.

Nous proposons d'illustrer ces évaluations à travers les exemples de MessagEase et ShapeWriter.

III.1. Evaluation de MessagEase

Nous considérons ici la version « complète » de MessagEase[15] et non la version expérimentale réduite aux 26 caractères de l'alphabet latin plus l'espace. Cette version offre plusieurs layouts permettant de saisir l'ensemble des caractères y compris les caractères spéciaux. Néanmoins, l'accès à ces différents layouts n'est pas immédiat, ni très intuitif et doit être appris. Par ailleurs, MessagEase est pensé pour l'anglais et l'accès aux caractères accentués s'avère complexe.

MessagEase offre également un raccourci pour faciliter la saisie des caractères majuscules (l'interaction de saisie, simple clic ou trait, est prolongée en encerclant la lettre).

III.1.a. Evaluation par rapport aux objectifs de saisie

	SMS Informel	SMS Formel	Email	Documents formatés
Facilité d'accès aux caractères spéciaux	0 Non requis pour les SMS informels	1 Accès indirect à différents layouts	1	0 Caractère plus fréquemment requis
Facilité d'accès aux caractères accentués	0 Non requis pour les SMS informels	-2 Accès complexe aux caractères accentués	-2	-2

[15] Un tutorial présenté ici illustre l'ensemble des possibilités de la version complète de MessageEase http://www.exideas.com/ME/tutorial/parent.swf

Facilité d'accès aux principaux caractères de ponctuation	2 Caractères accessibles directement comme de simples lettres	2	2	2
Facilité d'accès aux majuscules	2 Raccourci permettant de saisir les majuscules	2	2	2
Saisie 'intuitive' des majuscules	0 Non disponible	0	0	0
Complétion	0 Pas de complétion	0	0	0
Outil de désambiguïsation linguistique pour les claviers ambigus	0 Pas d'outil de désambiguïsation	0	0	0
Apprentissage du système de complétion sur la base des saisies utilisateur	0	0	0	0
	4	3	3	2

Ainsi, bien que MessagEase offre un accès à l'ensemble des caractères, il semble rester plus adapté à la saisie de textes simples sans grandes exigences de formatage.

III.1.b. Evaluation par rapport à la population

	Utilisateur lambda familier de AZERTY			Utilisateur lambda non familier de AZERTY			Personnes âgées			Handicapés moteurs		
	Fréq. d'utilisation			Fréq.			Fréq.			Fréq.		
	Haute	Moy.	Faible	H	M	F	H	M	F	H	M	F
Ordonnancement des caractères	-1	-2	-3	-1	-2	-3	-1	-2	-3	-1	-2	-3
	Le layout contre intuitif, basé sur aucune référence commune, est un gros handicap à l'utilisation du clavier particulièrement pour un usage peu fréquent.											
Nature de l'interaction	1	0	-1	1	0	-1	0	-1	-2	-2	-2	-3
	L'interaction différente entre la saisie des caractères principaux et secondaires est cognitivement plus complexe que l'usage de simples clics. Elle nécessite un temps d'adaptation et une certaine fréquence d'usage pour être naturelle. L'interaction est en outre physiquement difficile à exécuter pour des utilisateurs handicapés.											
Aides dynamiques additionnelles	0	0	0	0	0	0	0	0	0	0	0	0
	Pas d'aides dynamiques additionnelles											
Réagencements dynamiques	0	0	0	0	0	0	0	0	0	0	0	0
	Pas de réagencements dynamiques											
Taille des touches	2	2	2	2	2	2	2	2	2	2	2	2
	La taille des touches facilite le pointage, notamment des utilisateurs moins habiles tels que personnes âgées et handicapés.											
Réduction des distances parcourues par le pointeur	0	0	0	0	0	0	0	0	0	0	0	0
	Pas de disposition particulière pour réduire les distances parcourues par le pointeur											
	2	0	-2	2	0	-2	1	-1	-3	-1	-2	-4

La nature de l'interaction et la complexité du layout rendent le clavier particulièrement inadapté à un usage occasionnel. En outre, la nature de l'interaction est d'autant plus pénalisante que l'utilisateur est moins mobile des membres supérieurs.

III.1.c. Evaluation par rapport au contexte d'utilisation

	Usage simple	Usage en transport collectif	Usage en contexte de marche	Usage parallèle à une activité principale	Usage aveugle
Taille des touches	2	2	2	2	2
Aide dynamique en marge du clavier	0	0	0	0	0
Aide dynamique sur le clavier	0	0	0	0	0
Autocorrection d'erreurs	0	0	0	0	0
Agencement statique des caractères	0	0	-1	-3	-3
Complexité de l'interaction	0	0	0	-2	-3
	2	2	1	-3	-4

De par l'effort cognitif nécessaire à l'interaction et la complexité du layout, le clavier nécessite une attention particulière qui le rend inefficace lorsque la saisie est une tâche secondaire. En revanche, la taille des touches et le fait que l'interaction ne doit pas nécessairement être très précise le rendent assez robuste aux situations de mobilité.

III.1.d. Bilan

Ainsi, si nous considérons une tâche de saisie de SMS informels, par un utilisateur lambda utilisant fréquemment le clavier, dans une condition quelconque ou de mobilité, le clavier reste relativement bien adapté (note de 4+2+2=8) malgré une distribution de touches complexe à appréhender qui le pénaliserait fortement en comparaison à DashKey (qui, pour les mêmes propriétés techniques, propose une distribution de touches plus intuitive).

En revanche, le clavier est mal adapté à un usage ponctuel (toujours en raison de la distribution de touches) et mal adapté à des usagers aux capacités motrices limitées (telles que les personnes âgées et les utilisateurs handicapés moteur).

III.2. Evaluation de ShapeWriter

La dernière version de ShapeWriter [Zhai 09] (cf. Figure 20), permet un usage traditionnel du clavier en pointant un à un les caractères, mais il permet également la saisie des caractères de manière continue sans lever le pointeur. Sur la base de l'analyse des courbes et de la dynamique du tracé, un système de désambiguïsation permet de transformer les courbes ainsi dessinées en une succession de pointages. Cette succession de pointage est ensuite confrontée à un dictionnaire permettant de corriger les imprécisions de l'utilisateur et les ambiguïtés du tracé. Si l'ambiguïté ne peut pas être levée, le système propose une liste de complétion permettant de choisir entre les alternatives possibles.

Figure 20. ShapeWriter

III.2.a. Evaluation par rapport aux objectifs de saisie

	SMS Informel	SMS Formel	Email	Documents formatés
Facilité d'accès aux caractères spéciaux	0 (non requis pour les SMS informels)	0 Pas de stratégie particulière pour faciliter cette saisie	0 Idem	0 Idem
Facilité d'accès aux caractères accentués	0	-1 Le système est conçu pour l'anglais et n'offre pas de solutions directes pour la saisie des caractères accentués	-2 Idem	-2 Idem
Facilité d'accès aux principaux caractères de ponctuation	2 Caractères accessibles directement comme de simples lettres	2 Idem	2 Idem	2 Idem
Facilité d'accès aux majuscules	0 Pas de stratégie particulière	0 Idem	0 Idem	0 Idem
Saisie 'intuitive' des	0	0	0	0

majuscules	Pas de stratégie particulière			
Complétion	0 Le système de complétion a pour finalité de corriger les ambiguïtés introduites par le système. Il ne permet pas de faciliter la saisie par rapport à un outil normal	0	0	0
Outil de désambiguïsation linguistique pour les claviers ambigus	1 Le système de désambiguïsation permet de réduire le nombre et l'exigence en matière de pointage. En revanche il nécessite parfois d'être complété par une interaction avec la liste de complétion	1 Idem	1 Idem	1 Idem
Apprentissage du système de complétion sur la base des saisies de l'utilisateur	0	0	0	0
	3	2	1	1

Principalement parce qu'il est initialement conçu pour l'anglais, qui ne contient pas de caractères accentués, le clavier n'est pas idéalement adapté pour la saisie de documents formels.

III.2.b. Evaluation par rapport à la population

	Utilisateur lambda familier de AZERTY			Utilisateur lambda non familier de AZERTY			Personnes âgées			Handicapés moteurs		
	Fréq. d'utilisation			Fréq.			Fréq.			Fréq.		
	Haute	Moy.	Faible	H	M	F	H	M	F	H	M	F
Ordonnancement des caractères	2	2	2	0	-1	-2	0	-1	-2	2	2	2
	ShapeWriter utilise les ordonnancements de caractères standards (QWERTY / AZERTY etc.) pertinents pour des utilisateurs familiers avec l'environnement informatique											
Nature de l'interaction	2	1	1	2	1	1	2	2	1	0	0	0
	Le tracé, corrigé par un système de désambiguïsation, est moins exigeant en termes de précision. L'interaction devrait donc convenir assez bien à des utilisateurs moins précis dans leurs gestes. En revanche le tracé continu est assez inapproprié aux utilisateurs handicapés moteurs qui peuvent néanmoins utiliser le clavier comme un simple AZERTY. L'utilisation de la liste de complétion a néanmoins un coût qui est plus préjudiciable pour les utilisateurs les plus rapides											
Aides dynamiques additionnelles	0	0	0	0	0	0	0	0	0	0	0	0
Réagencements dynamiques	0	0	0	0	0	0	0	0	0	0	0	0
Taille des touches	1	1	1	1	1	1	1	1	1	0	0	0
	La tolérance dans la précision du geste continu est équivalente à une taille de touche supérieure. Pour les utilisateurs handicapés moteurs utilisant le clavier comme un clavier standard, cette aide n'est d'aucun support.											
Réduction des	0	0	0	0	0	0	0	0	0	0	0	0

distances parcourues par le pointeur												
	5	4	4	3	1	0	3	2	0	2	2	2

La distribution des caractères basée sur les distributions standard (QWERTY/AZERTY etc.) est pertinente pour la majorité des utilisateurs en contact régulier avec l'environnement informatique. L'interaction, simple et tolérante aux imprécisions, est adaptée à l'ensemble des utilisateurs en dehors des utilisateurs handicapés moteur qui peuvent néanmoins utiliser le clavier comme un clavier standard.

III.2.c. Evaluation par rapport au contexte d'utilisation

	Usage simple	Usage en transport collectif	Usage en contexte de marche	Usage parallèle à une activité principale	Usage aveugle
Taille des touches	0	1 L'imprécision tolérée dans le pointage peut être d'une aide substantielle en situation de mobilité	1 idem	0	-1 L'interaction n'est cependant pas suffisamment permissive pour permettre un usage aveugle efficace
Aide dynamique en marge du clavier	0	0	0	-1 L'usage nécessaire de la liste de complétion requiert une certaine attention	-2
Aide dynamique sur le clavier	0	0	0	0	0
Autocorrection d'erreurs	1 Le système de désambiguïsation	1 idem	1 idem	1 idem	1 idem

	permet de réduire les erreurs				
Agencement statique des caractères	0	0	0	0	0
Complexité de l'interaction	0	0	0	0	0
	1	2	2	0	-2

La tolérance aux imprécisions dans le pointage rend le clavier particulièrement intéressant en situation de mobilité.

III.2.d. Bilan

Le clavier s'avère donc bien adapté particulièrement pour la saisie de SMS informels, en contexte de mobilité, pour des utilisateurs aguerris au clavier AZERTY/QWERTY. Sa conception pour l'anglais (ne prévoyant pas un accès direct aux accents) le rend en revanche mal adapté à la saisie de textes formels en français. Le clavier peut en outre être utilisé comme un simple AZERTY ce qui le rend facile d'approche et accessible aux utilisateurs handicapés (qui ne profiteront néanmoins pas des avantages du système).

Section IV - Conclusion

De manière à offrir une base de comparaison plus représentative des perspectives d'utilisation des claviers, dans un cadre d'usage réel, que les simples performances maximales atteignables avec le clavier, nous avons proposé une grille d'évaluation support à une évaluation heuristique des claviers. Cette grille tient cependant uniquement compte de l'adéquation des principes de fonctionnement du clavier à un but, une population et un contexte d'utilisation. Cette grille ne compare pas d'éventuels éléments de design graphique par exemple. En outre l'évaluation des différents critères reste à la discrétion des évaluateurs experts.

Mais, bien que les points attribués à chaque critère par l'évaluateur puissent être sujet à discussion, cette évaluation permet d'offrir une estimation plus riche et plus nuancée que l'établissement d'une simple valeur de référence à l'objectivité contestable.

Conclusion

A travers l'étude des différentes méthodologies d'évaluation théoriques et expérimentales des claviers logiciels, nous avons observé les limites actuelles de la validité de celles-ci. Outre l'absence de significativité de la notion d'utilisateur novice, nous avons mis en évidence l'ambiguïté entre la notion d'utilisateur expert et la notion d'utilisateur parfait qui remet particulièrement en cause la mesure et la comparaison des performances pour les claviers complexes (proposant notamment des aides dynamiques).

Notre initiative pour améliorer l'estimation théorique des performances des utilisateurs débutant avec un nouveau clavier, nous a permis de mettre en évidence la relativité de ces performances à un héritage « culturel » de l'utilisateur. Nous avons en conséquence identifié que, au-delà d'un calcul formel, ces résultats pouvaient être obtenus intuitivement sur la base de quelques heuristiques concernant les connaissances de l'utilisateur préalables à l'utilisation du clavier.

Nous avons en conséquence montré deux aspects : en premier lieu les claviers ne peuvent pas être comparés de façon absolue sur la base d'une simple valeur numérique ; en second lieu les perspectives d'utilisation d'un clavier par une population sont relatives aux acquis de cette population et, plus largement, à la fréquence et au contexte d'utilisation de l'artefact ainsi qu'au but même de l'utilisateur (en d'autres termes, la qualité du texte saisi en matière de respect des normes linguistiques et en matière de formatage).

En conséquence, nous avons proposé une grille d'évaluation heuristique permettant d'accompagner l'estimation et les perspectives d'utilisation des claviers en fonction de certaines valeurs caractéristiques pour chacune des variables : population, fréquence d'utilisation, contexte d'utilisation et but. Cette grille n'a pas pour objectif d'effacer définitivement les évaluations expérimentales (voire, sous réserve les évaluations théoriques), mais elle doit permettre de définir de façon plus nuancée les conditions d'utilisation du clavier qui facilitent ou non son utilisation et son adoption par une population.

Deuxième Partie

Encadrement de l’évaluation et du design des claviers logiciels

---

Comme nous avons pu l'observer dans la première partie, une évaluation objective des claviers logiciels exige la confrontation de plusieurs données qualitatives et expérimentales relatives à différentes considérations sur l’usage, incluant le profil de l'utilisateur et l’environnement d’utilisation. Ces évaluations restent laborieuses à mettre en œuvre et nos travaux se sont, en conséquence, dirigés vers l'élaboration d'un outil d’instrumentalisation de celles-ci.

Les travaux initialement conduits par Mathieu Raynal [Raynal 05d], avaient abouti à la création d'une première version de la plate-forme d’expérimentation des claviers logiciels : E-Assiste [Raynal 05c, Raynal 05d]. Cette plate-forme proposait un ensemble de briques logicielles permettant de monter rapidement différentes expérimentations autour des claviers, et de comparer un ensemble de claviers sur la base d’un protocole expérimental commun. Le montage d'une expérimentation exigeait cependant un assemblage par programmation de ces briques logicielles. La plate-forme était en outre exclusivement à destination d’un environnement bureautique, ne permettant notamment pas d’effectuer des évaluations sur les dispositifs mobiles tels que téléphones portables.

Nous aborderons dans cette partie les orientations que nous avons prises pour proposer une seconde version de cette plateforme afin de la faire évoluer vers un outil complet et multi plate-formes d'étude des claviers logiciels. Ces évolutions sont articulées autour de quatre principaux axes de travail.

Dans un premier chapitre (Chapitre 6), nous évoquerons les travaux initiaux concernant la plate-forme E-Assiste ainsi que d'autres travaux visant à l'instrumentalisation de l'étude des claviers logiciels.

L’objectif initial de la plateforme consiste en la mise en œuvre d’expérimentations reproductibles. Notre première démarche, exposée dans le chapitre 7, a été en conséquence de modéliser et formaliser le déroulement d’une expérimentation. En lieu et place d’un assemblage manuel des briques logicielles fournies par la plateforme, la nouvelle plateforme – E-Assist II – propose de décrire le protocole expérimental (en XML) : claviers, consignes, exercices, groupes d’utilisateurs et déroulement des sessions en fonction de ces groupes. La plateforme gère ensuite de façon automatique l’adhésion des utilisateurs aux groupes, le déroulement des sessions et fournit des outils d’analyses pour extraire les principaux résultats et étudier leur validité (à travers un ANOVA).

Comme nous l'avons préalablement évoqué dans la première partie, les claviers ne sont plus aujourd'hui de simples jeux de boutons aux formes plus ou moins variées et aux distributions de touches diverses. Ils intègrent non seulement d'autres paradigmes d'interaction que la simple pression d'un bouton, mais également des outils de traitement linguistique qui vont corriger la saisie, suggérer des raccourcis ou encore altérer la représentation du clavier. A travers l'étude des claviers, nous avons identifié une architecture commune entre la grande majorité de ceux-ci, et structuré cette architecture autour de plusieurs blocs fonctionnels. Sur la base de ces blocs fonctionnels et de leur interaction, nous proposerons donc, dans un deuxième temps, un langage – KeySpec – destiné à créer des claviers, ainsi qu'un ensemble de briques logicielles (un SDK), destiné à compiler et exécuter ce langage (cf. Chapitre 8).

De manière à avoir des données de comparaison rapides sur les claviers sans passer systématiquement par une expérimentation (et ce malgré la relativité de ces informations discutées en partie 1), et de simuler ainsi l'impact de différents paramètres sur la saisie de texte par rapport aux modèles prédictifs, nous avons, dans un troisième temps (Chapitre 9), intégré la possibilité de spécifier les données relatives aux modèles prédictifs dans le langage de génération de clavier. La plate-forme E-Assist II a été consécutivement munie d'un outil de simulation de saisie de texte exploitant les données fournies et générant automatiquement les évaluations théoriques relatives aux claviers spécifiés en KeySpec.

Par ailleurs, de manière à compléter les évaluations des claviers logiciels en intégrant l'étude heuristique, nous avons proposé un outil (cf. Chapitre 9) permettant d'encadrer le déroulement de cette étude heuristique par rapport aux différentes conditions (objectif, population, fréquences d'utilisation du clavier et contexte d'utilisation).

Chapitre 5 - Instrumentalisation de l'évaluation des claviers logiciels

Plusieurs travaux existants abordent la question de l'instrumentalisation de l'évaluation des claviers logiciels. Ces travaux développent trois axes de recherche :

- L'encadrement des expérimentations (cf. Section I). Ces travaux visent à concevoir des outils pour faciliter la mise en œuvre et l'encadrement d'expérimentations. C'est le cas de la plate-forme E-Assiste [Raynal 05c], de [Poirier 08], de TextTest [Wobbrock 06b], TimTester[16] [Martin 08, Martin 09] et, d'en un contexte plus restrictif, également de celle de SoKeyTo [Vella 04, Vella 05] ;

- L'évaluation théorique des claviers logiciels (cf. Section II). Ces travaux visent à fournir des outils permettant de modéliser les claviers et de déduire automatiquement, à partir de ces modèles, des données concernant les performances théoriques des claviers. C'est le cas des travaux de Castellucci et MacKenzie [Castellucci 09] ;

- La conception des claviers logiciels (cf. Section III). Ces travaux visent à fournir des outils permettant de faciliter la conception et la génération des claviers logiciels. C'est principalement le cas des travaux autour de SoKeyTo [Vella 04, Vella 05].

Nous expliciterons les intérêts et limites des travaux menés dans le cadre de ces trois démarches et justifierons l'intérêt d'une plate-forme améliorant *a minima* ces trois aspects (cf. Section IV).

Section I - Encadrement des expérimentations: E-Assiste, TextTest, TimTester

Raynal [Raynal 05c, Raynal 05d] observe la grande variabilité des protocoles d'expérimentations utilisés pour évaluer les claviers logiciels. Il en déduit que, si ces différences se justifient parfois, notamment autour de la tâche observée, les différentes métriques utilisées et surtout les différentes manières de les calculer compliquent considérablement une comparaison objective des artefacts.

En réponse, à travers la plate-forme E-Assiste, il vise à normaliser la conception des expérimentations. Dans cette perspective, il propose un ensemble de briques logicielles, articulées autour d'un protocole commun de communication.

Plus concrètement il fournit :

- Le support à la réalisation d'un ensemble de tâches matérialisées par des bandeaux de présentation de l'information, autorisant, par exemple, différents types d'exercices tels que la recopie de mots ou de textes ;

- Un ensemble de claviers développés pour les besoins des expérimentations. A noter qu'il n'est pas nécessaire, comme nous verrons ultérieurement, de redévelopper un clavier pour qu'il puisse être utilisé par la plate-forme. Des claviers fournis par

[16] http://www.cs.uta.fi/~poika/timtester/timtester.jar

d'autres « prestataires » et éventuellement conçus dans des langages différents de celui de la plate-forme (Java) pourront être, moyennant quelques adaptations, intégrés à la plate-forme ;

- Des outils permettant d'enregistrer les métriques nécessaires à l'évaluation des claviers : vitesse de saisie, erreurs et distance parcourue par le dispositif de pointage, performance des systèmes de prédiction ;

- Un bandeau permettant de présenter des consignes aux utilisateurs.

- Et enfin, des outils d'analyse d'une session d'expérimentation montrant : les performances mesurées, l'évolution dans le temps des performances sous forme d'histogramme et, enfin, les tracés effectués par le pointeur au cours de la saisie.

Ces différentes briques communiquent à travers le bus Ivy [Buisson 02, Merlin 04]. Le bus Ivy est un bus logiciel comparable à CORBA, mais plus léger et plus simple d'utilisation. Il permet à différents modules ou applications de communiquer par l'échange de messages textuels à travers un réseau Ethernet. Les claviers, éventuellement extérieurs à la plate-forme, peuvent ainsi communiquer avec les bandeaux de présentation de l'information en émettant des messages sur le bus Ivy. Les métriques nécessaires à l'évaluation des claviers sont acquises par la capture des messages émis sur ce même bus (cf. Figure 21).

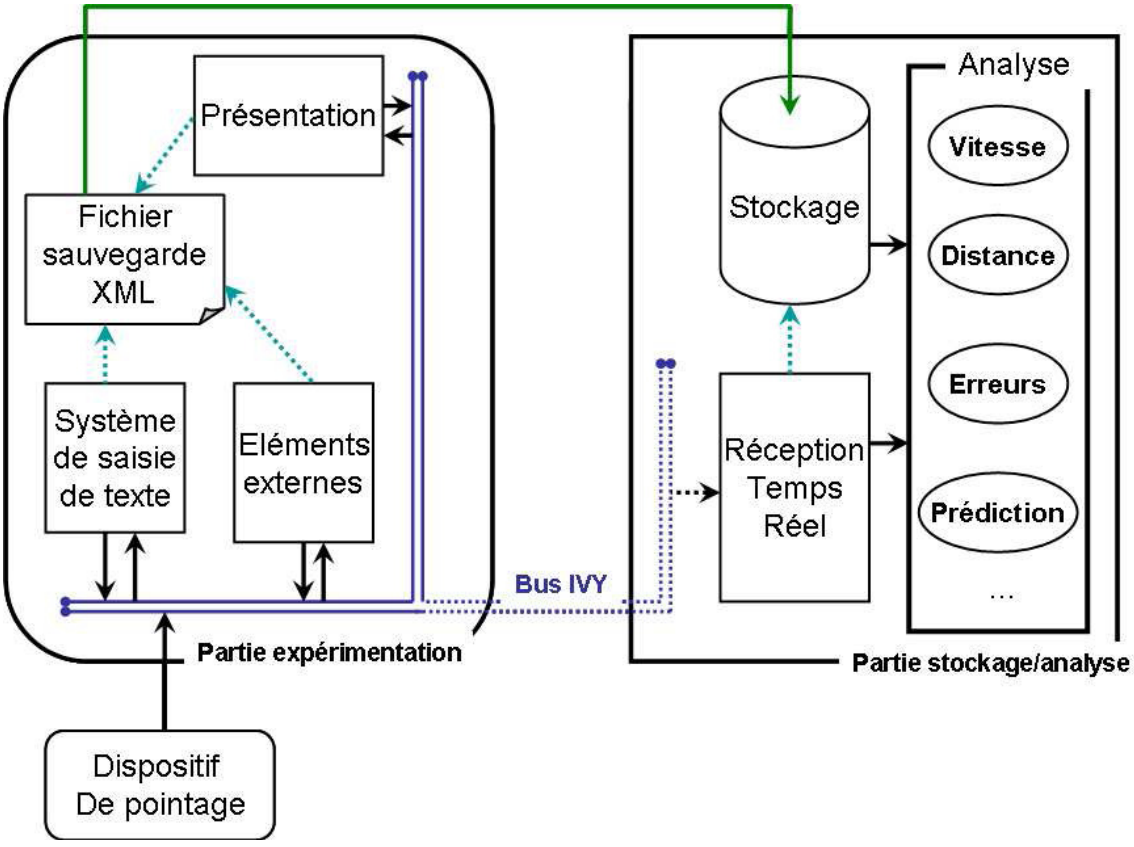


Figure 21. Architecture de la plateforme E-Assiste [Raynal 05d]

Ainsi, une session d'expérimentation est matérialisée par une succession d'instances de ces briques logicielles. Par exemple : une succession de bandeaux de consignes ; suivi de la co-présence, durant le déroulement d'un exercice, d'un clavier, d'un bandeau de présentation

des données de l'exercice et du module d'acquisition des résultats ; suivi éventuellement d'une temporisation, puis de nouvelles consignes et de nouveaux exercices jusqu'à ce que les différents exercices composant la session soient complétés.

Les scénarios d'expérimentation sont élaborés à travers une étape de programmation. Plusieurs scénarios peuvent être définis pour une même expérimentation et correspondent, par exemple, à différentes sessions pour des groupes éventuellement différents d'utilisateurs. Au cours de l'expérimentation, une fenêtre de login permet de choisir le scénario de session qui va être exécuté.

Ainsi, la plate-forme offre un support pour faciliter la mise en place d'une expérimentation. Néanmoins, dans sa version initiale, elle présente certaines limitations. En premier lieu, la nécessité d'une étape de programmation requiert la présence d'un développeur. Elle ne peut donc pas être manipulée par un expérimentateur quelconque. Par ailleurs, elle ne permet pas d'encadrer de façon déterministe une expérimentation composée de plusieurs sessions. Il est donc nécessaire de gérer manuellement à la fois l'attribution des scénarios pour chaque session, mais également le recueil des résultats. Ceci peut s'avérer être une source d'erreurs au cours du déroulement du protocole d'expérimentation. D'une manière plus générale concernant le traitement des résultats, si elle offre un support pour le traitement informatique pour une session d'expérimentation, la plate-forme n'offre en revanche pas d'outils (ANOVA par exemple) pour traiter collectivement l'ensemble des résultats et vérifier leur validité.

Par ailleurs, TextTest [Wobbrock 06b] et TimTester [Martin 08] sont deux autres outils permettant le recueil et l'analyse de l'interaction d'un utilisateur avec différents claviers. TextTest se dédie plus spécifiquement au traitement des expérimentations dont la tâche de saisie ne contraint pas pas l'utilisateur à suivre le texte présenté. Il offre en conséquence des outils adaptés pour étudier les erreurs dans cet environnement non contraint. TimTester est une alternative compatible avec les outils d'analyse fournit par TextTest. TimTester corrige par ailleurs une lacune de TextTest qui ignore, dans la mesure des erreurs, l'activité résultant à déplacer le curseur.

Section II - Outils de d'évaluation théorique des claviers logiciels

II.1. TnToolkit

Les travaux de Castellucci [Castellucci 09] s'inscrivent dans la continuité des travaux de MacKenzie traitant de l'exploitation des modèles prédictifs dans le but d'évaluer et comparer les claviers logiciels. Castellucci propose une toolkit (TnToolkit) permettant de faciliter la mise en œuvre du calcul prédictif essentiellement pour une catégorie particulière de claviers ambigus (basés sur l'algorithme de désambiguïsation T9).

La TnToolkit fournit un certain nombre d'outils permettant la spécification des caractéristiques du clavier sur la base d'une simple image du clavier. Un outil de sélection graphique (cf. Figure 22) permet, par exemple, de délimiter l'espace des touches. Un second outil permet d'attribuer à chaque touche ainsi délimitée un jeu de caractères (cf. Figure 23) ainsi que le doigt utilisé dans le cas d'une utilisation à deux pouces du clavier.

Les données géométriques ainsi saisies seront ensuite utilisées pour spécifier la difficulté de la cible dans les calculs relatifs à la loi de Fitts. De même, la disposition des caractères, associée aux données géométriques, fournit les éléments nécessaires pour le calcul relatif au model de Soukoreff et MacKenzie (cf. Chapitre 1 - II.1) et l'établissement du *lower-bound* et du *upper-bound*.

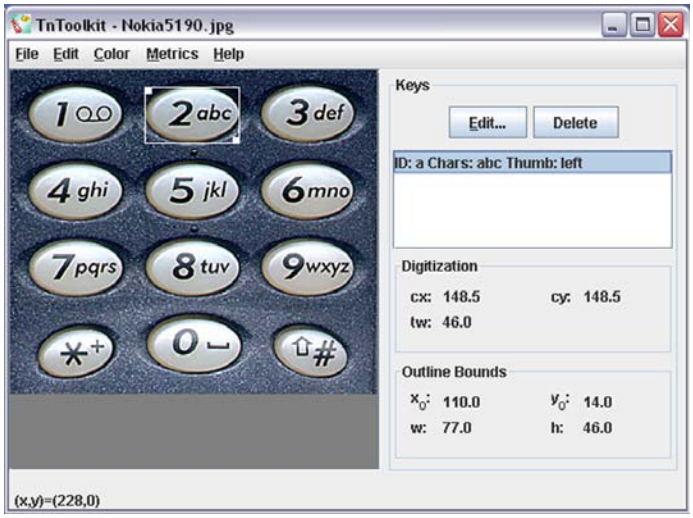


Figure 22. Désignation de la géométrie des touches à partir d'une image

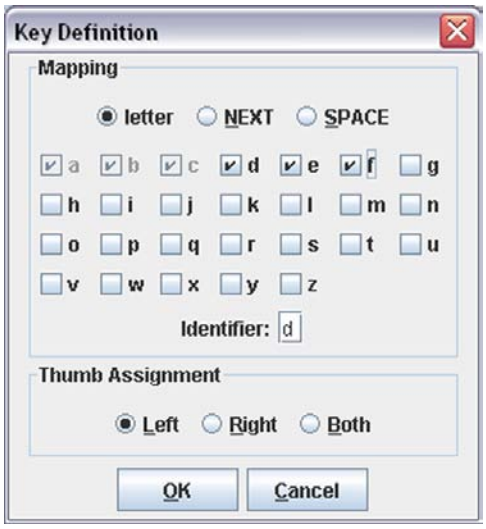


Figure 23. Affectation des caractères aux touches

En ce qui concerne l'obtention des performances en mots par minute (WPM), le calcul est basé sur le modèle prédictif de Soukoreff et MacKenzie [Soukoreff 95] corrigé pour la saisie à deux pouces [MacKenzie 02a]. En ce qui concerne l'obtention du nombre de frappes par caractère (KSPC, cf.Chapitre 1 - III.2.c), le calcul est basé sur les travaux de MacKenzie, [MacKenzie 02b].

L'algorithme de désambiguïsation de la sélection des caractères considéré est le T9. Les performances sont en conséquence calculées par simulation de la saisie des mots d'un dictionnaire et pondérées par la fréquence des mots dans la langue considérée. La liste de fréquence des mots est fournie en entrée du dispositif de simulation. La simulation de la pression d'une touche « NEXT » (affectée de la même manière que sont affectés les caractères) permet de désambiguïser la saisie dans les cas où l'algorithme T9 ne propose pas le mot adéquat en premier.

La toolkit ne permet pas de spécifier d'autres méthodes de désambiguïsation de la saisie telles que LetterWise [MacKenzie 01], ni la désambiguïsation par interaction [Nesbat 03], ou

même le mode MultiTap traditionnel. En conséquence, la TnToolkit permet de comparer essentiellement une gamme de claviers, basés sur le mode de fonctionnement du T9, et proposant éventuellement une géométrie et/ou une distribution différente de touches.

Elle pourrait également être utilisée avec des claviers simples tels que QWERTY, Metropolis [Zhai 00], GAG [Raynal 05b], etc. En revanche, elle ne permet pas de spécifier des mécanismes d'interaction complexes relatifs à un système de prédiction (ne serait-ce que modéliser plusieurs items d'une liste de complétion accessibles simultanément par pointage).

Par ailleurs, le modèle prédictif utilisé exploite les données fournies par le système de prédiction du T9 sans introduire de coût cognitif relatif à l'usage des résultats de la prédiction. Comme nous l'avons évoqué précédemment, les résultats générés par l'outil, en ce qui concerne la vitesse de saisie, correspondent donc aux performances d'un utilisateur « parfait » du système (capable d'anticiper, pour chaque mot, les résultats du système de prédiction).

II.2. Calcul du KSPC à partir de Machines à états finis

Sandnes [Sandnes 05] étudie les différentes techniques de saisie sur claviers ambigus tel que Multitap, Wristwatch [Raghunath 02], Multi-Ring [Sandnes 04], etc. Pour les différentes techniques étudiées, la sélection d'un caractère est désambigüisée par une succession de plusieurs actions, à l'exemple de MultiTap ou pour saisir le premier caractère présent sur une touche il faut presser la touche une fois, pour saisir le second, deux fois, etc. En outre, il observe que, pour des claviers tel que Chord Keyboard [Rochester 78, Wigdor 04], par exemple, le signal déclenchant la saisie n'est pas nécessairement une frappe de touche. De même, pour MultiTap, le dernier signal peut être un simple délai.

En conséquence, il propose de représenter les différentes techniques sous la forme d'automates finis (cf. Figure 24, Figure 25, Figure 26, Figure 27, Figure 28) dont : les transitions sont activées par un signal ; les états sont des situations pas nécessairement perceptibles graphiquement ; les états sortants notifient le système de la saisie d'un caractère.

Cette représentation permet d'une part de matérialiser de manière visuelle le comportement des claviers ambigus, mais les parcours dans ces automates permettent surtout de déterminer automatiquement le KSPC de chaque clavier, ainsi que le coût du traitement de récupération en cas d'erreur de l'utilisateur.

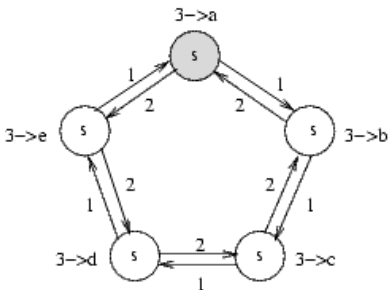


Figure 24. Classic data-stamp

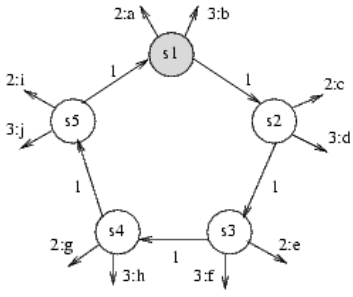


Figure 25. Data-stamp

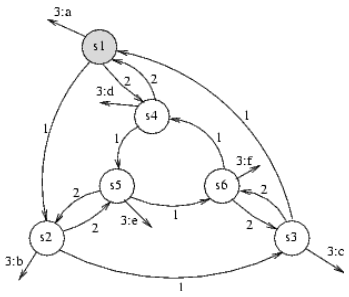


Figure 26. Wristwatch

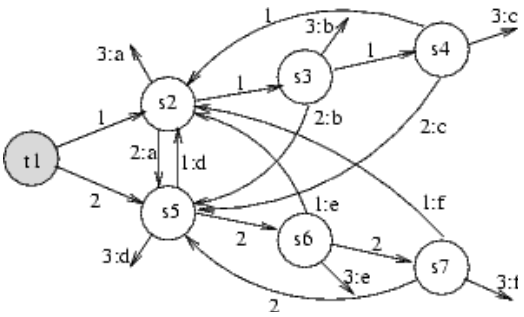


Figure 27. Multi-tap

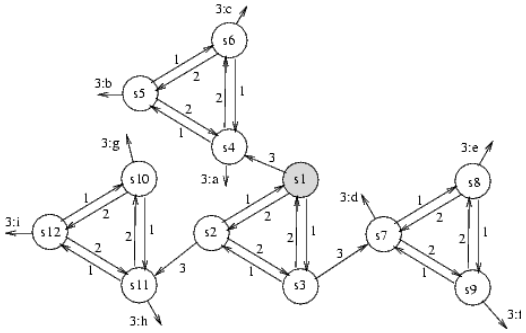


Figure 28. Multi-ring

Néanmoins, cet outil de modélisation n'incorpore pas la possibilité de comparer des claviers dont la frappe est désambiguïsée par l'intermédiaire de systèmes de prédiction tels que le T9[17], LetterWise, etc., ou encore désambiguïsée par une interaction plus complexe telle un geste comme dans les cas de MessagEase ou DashKey.

Section III - Outils de conception des claviers logiciels : SoKeyTo

La dernière orientation, proposée par Vella [Vella 04, Vella 05], est de fournir un outil de conception des claviers logiciels. Au lieu de partir d'un outil existant et de fournir des outils pour le décrire, comme dans le cadre de la TnToolkit, SoKeyTo permet de décrire le clavier et de le générer à partir de cette description.

Cet manière d'aborder présente deux avantages. Elle permet, en premier lieu, de fournir un outil de spécification manipulable par un utilisateur final. En effet, l'IHM, sur la base de simples drag&drop et copier/coller, permet l'élaboration de nouvelles distributions de touches et, par exemple, d'additionner des raccourcis claviers vers des mots fréquemment utilisés par l'utilisateur. En second lieu, la plate-forme tient lieu de support à une évaluation théorique (sur la base de l'application du modèle de Soukoreff et MacKenzie [Soukoreff 95]) et expérimentale des claviers ainsi conçus (à l'exemple des claviers Annie et Céline générés au cours des évaluations du prototype, cf. Figure 29).

Néanmoins la plate-forme de conception aborde exclusivement la problématique de la disposition des caractères. En conséquence, elle ne permet pas de spécifier des touches ambiguës, ou des modes alternatifs d'interaction. Elle ne permet pas non plus l'interaction avec des systèmes de prédiction et d'éventuelles altérations dynamiques du clavier. Les claviers générés sont donc composés d'un layout unique.

[17] Reduced keyboard disambiguating system, Patent -US6307549, (1995).

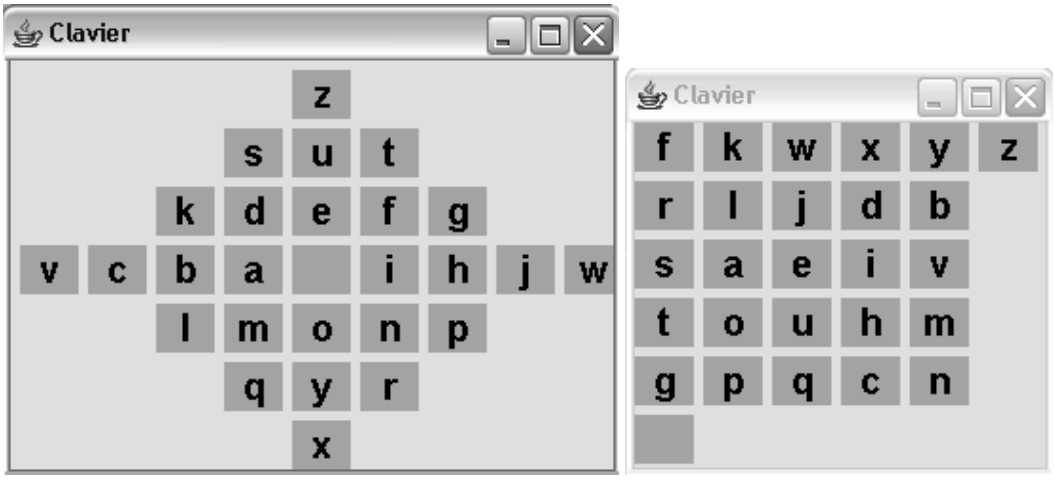


Figure 29. Clavier Annie (gauche) et Céline (droite)

Section IV - Conclusion

Ainsi, les différents projets traitant de l'instrumentalisation de l'évaluation des claviers logiciels abordent trois aspects différents de cette évaluation : modélisation et encadrement de l'expérimentation, modélisation des claviers dans la perspective d'évaluations théoriques, et modélisation des claviers dans la perspective d'évaluations expérimentales. Mais, comme nous avons pu le constater, les outils de modélisation s'attaquent chacun à un sous-ensemble assez restreint de claviers.

En outre, aucun des outils ne permet d'évaluer un clavier de manière plus large en pondérant les résultats obtenus par des données relatives à un objectif, une population cible ou/et un contexte d'utilisation.

L'objectif des travaux menés du point de vue de l'instrumentalisation de l'évaluation des claviers logiciels, dans le cadre de cette thèse, est de poursuivre ces trois axes déjà préalablement abordés et d'aborder ce quatrième. Ils viseront notamment à étendre les possibilités en matière de modélisation des claviers logiciels (à des fins de génération de claviers ou d'évaluation prédictive), à améliorer la formalisation et l'encadrement des protocoles d'expérimentation, et surtout à synthétiser ces différents axes au sein d'une même plate-forme de conception et d'évaluation des claviers logiciels : E-Assist II.

Chapitre 6 - Formalisation des protocoles d'expérimentation

Comme nous l'avons évoqué préalablement, l'idée de la première plate-forme E-Assiste était de fournir un ensemble de briques logicielles permettant de monter une expérimentation. Ces briques logicielles partageaient une architecture commune permettant leur interaction et offraient des services tels que : bandeau de consignes ; bandeaux de présentation de l'information à recopier ; et claviers. Une fois ces briques assemblées par une étape de programmation, la plate-forme permettait d'accompagner un utilisateur au cours d'une session d'expérimentation et de recueillir les résultats bruts de la session.

Néanmoins, comme nous l'observerons à travers l'étude de différentes expérimentations sur des claviers logiciels (cf. Section I), les expérimentations sont souvent multisessions et requièrent différents groupes d'utilisateurs. En conséquence, nous expliciterons les travaux que nous avons menés (cf. Section II) pour faciliter l'intégration des briques logicielles et fournir une meilleure gestion des expérimentations. De plus, dans la mesure où un clavier peut être sensible au support (Tablet PC, Palm, et différents types de téléphone) et de manière à étendre les moyens d'expérimentation à la majorité des dispositifs, nous avons proposé une version J2ME de la plate-forme E-Assist II : TinyEAssist (cf. Section III).Nous discuterons enfin (cf. Section IV) des travaux qui pourraient être conduits pour améliorer ce fonctionnement dans une future version.

Section I - Besoins relatifs à l'encadrement des expérimentations

Comme nous l'avons évoqué dans la première partie, les expérimentations visant à évaluer les claviers logiciels sont généralement structurées autour de trois objectifs différents :

- soit l'objectif est une étude longitudinale des performances. L'expérimentation engage alors l'utilisateur dans une évaluation sur plusieurs sessions. La tâche de recopie exploite systématiquement le même clavier mais éventuellement des textes différents pour évaluer l'apprentissage du clavier sans l'apprentissage du texte ou, au contraire, des textes identiques pour mettre l'accent sur les performances lorsque les automatismes [Zhai 00, MacKenzie 99], y compris en relation au texte à saisir, sont acquis par l'utilisateur.

- soit l'objectif engage les utilisateurs dans une étude comparative de deux claviers [Magnien 04, Isokoski 04, Tanaka-ishii 07, Gong 08] ou plus [Vigouroux 04]. Dans ce contexte, les utilisateurs vont affronter une ou plusieurs sessions au cours desquelles ils vont effectuer plusieurs exercices au moyen des différents claviers. Les exercices vont potentiellement être alternés au cours des sessions de manière à permettre le contrebalancement des effets d'apprentissage et de fatigue [Raynal 05c]. A cette même fin, les utilisateurs sont en général répartis en différents groupes chacun initiant la première session par un exercice différent, le nombre de groupes correspond ainsi à la combinatoire des exercices. Une autre option consiste à ce que chacun des groupes effectue la ou les sessions avec un seul des dispositifs. On comptera donc le même nombre de groupes que de dispositifs.

- soit l'objectif peut être de comparer l'usage d'un ou plusieurs claviers en fonction de différentes populations [Raynal 07c].

Les expérimentations sur les claviers logiciels manipulent donc différents éléments nécessaires à leur organisation : les briques logicielles utiles à l'élaboration des exercices (bandeau de présentation des informations, claviers, outils de capture des résultats) ; les données textuelles pour les exercices ; les consignes fournies aux utilisateurs; les différents scénarios exploités par les différents groupes qui peuvent potentiellement être déduits du nombre de dispositifs testés et de l'une des deux stratégies préalablement citées ; et enfin les utilisateurs affectés aux différents groupes (pouvant éventuellement être affectés à un groupe en fonction d'une caractéristique spécifique, un handicap par exemple).

En réponse, les expérimentations fournissent un certain nombre de résultats numériques : moyennes, écarts types des temps et distances parcourues par le pointeur en fonction des sessions, des exercices, des utilisateurs ou des dispositifs ; mais également l'étude de la validité de ces résultats en fonction des différents paramètres de l'expérimentation et notamment, à travers l'analyse de la covariance (ANOVA) permettant l'étude de la significativité des résultats par rapport au nombre d'utilisateurs ou de la neutralité, entre les dispositifs, des effets d'apprentissage et de la fatigue.

L'objectif des premiers travaux effectués sur la plate-forme E-Assist II était donc de fournir un support destiné à orchestrer l'ensemble des éléments nécessaires à la conception de l'expérimentation, puis d'en extraire les variables et les valeurs requises pour l'analyse des résultats.

Section II - Formalisation des expérimentations

Nous avons en premier lieu proposé un langage XML de description des expérimentations (II.1). Ce langage de description devait nous permettre de suppléer, dans E-Assiste, la création des scénarios réalisés originellement par l'intermédiaire d'une étape de programmation. Ces informations sont par la suite traitées pour accompagner l'expérimentation (II.2) et organiser les résultats (II.3).

II.1. Langage de description des expérimentations

Le langage nous permet en premier lieu de définir des conditions (cf. Exemple 1) : un dispositif testé dans un contexte d'utilisation (exercice ou session d'apprentissage par exemple).

```
<!-- La balise XML « condition » nous permet d'associer un dispositif et une liste de
consignes spécifiques qui vont introduire l'usage du dispositif à chacune des instances
de la condition durant l'expérimentation. -->

<condition name="entrainement" keyboard="azerty" instructions="consignes_azerty.txt" />
```

Exemple 1. Définition d'une condition

En second lieu, le langage introduit la possibilité de déclarer des groupes d'utilisateurs et optionnellement des caractéristiques restreignant l'appartenance d'un utilisateur à ce groupe (cf. Exemple 2). Nous discuterons un peu plus loin comment sont traitées ces caractéristiques pour automatiser l'affectation des utilisateurs aux groupes (cf. Exemple 6).

```
<!-- la balise « group » spécifie les groupes d'utilisateurs qui vont participer à
l'expérimentation, et éventuellement les caractéristiques particulières des membres
participants de ce groupe -->
<group name="jeunes_handicapés" age="10-15" statut="handicapé" />
```

Exemple 2. Définition d'un groupe d'utilisateurs

Le langage va par ailleurs nous permettre de définir le contenu des sessions d'expérimentation, les différents scénarios d'expérimentation (exemple 4), et dans quel contexte tel ou tel scénario va être exécuté (cf. Exemple 3)

```
<!-- La balise « session » va nous permettre de matérialiser un scénario
d'expérimentation. Le scénario suivant sera exécuté pour le groupe « gp1 » au cours des
sessions 1 à 5 et pour le groupe « gp2 » au cours des sessions 1,3 et 5. -->
<session context="gp1:1-5;gp2:1,3,5" />
[…]
</session>
```

Exemple 3. Caractéristiques d'une session

Les différentes étapes du scénario (cf. Exemple 4) peuvent être : des consignes (balise « instructions ») passées sous forme de textes simples ou de fichiers HTML si nécessité, (par exemple pour présenter des images) ; des exercices (balise « exercise ») qui seront l' association d'une condition avec un texte et un mode de présentation de l'information ; ou un questionnaire (discuté un peu plus loin, cf. Exemple 8). Au cours d'un scénario, un exercice peut être exécuté conditionnellement à une session.

```
<!-- Dans l'exemple suivant, les scénarios sont initiés par deux bandeaux
d'instructions affichant les textes instruction-1.txt puis instruction-2.html, puis
trois exercices vont être exécuter (Dans le cas du premier scénario : l'exercice
répondant à la condition entraînement, puis l'exercice répondant à la condition azerty,
puis l'exercice répondant à la condition alpha). Dans les deux scénarios, le premier
exercice relatif à la condition « entrainement » est exécuté uniquement s'il s'agit de
la première session effectuée par l'utilisateur. -->
<session context="gp1:2,4,6;gp2:1,3,5" />
    <instruction texts="instruction-1.txt;instruction-2.html;" />
    <exercise session="1" condition="entrainement" text="mots_0" presentation="mots" />
    <exercise condition="azerty" text="mots_1" presentation="mots" />
    <exercise condition="alpha" text="mots_2" presentation="mots" />
</session>
<session context="gp2:2,4,6;gp1:1,3,5" />
    <instruction texts="instruction-1.txt;instruction-2.html;" />
    <exercise sessions="1" condition="entrainement" text=" mots_0" presentation="mots" />
    <exercise condition="alpha" text="mots_1" presentation="mots" />
    <exercise condition="azerty" text="mots_2" presentation="mots" />
</session>
```

Exemple 4. Définition du contenu des sessions

Enfin, l'expérimentation nécessite parfois de recueillir des données sur l'utilisateur, par exemple l'âge, l'aspect droitier ou gaucher, la nature valide ou handicapée du sujet, la catégorie socio-professionnelle, etc. Ces mêmes données peuvent être utilisées de manière à classifier un utilisateur parmi l'un des différents groupes lorsque l'expérimentation compare l'usage d'un artefact au sein de différentes populations. Le langage va donc nous permettre de spécifier les variables à recueillir lorsque l'utilisateur va adhérer à la plateforme (cf.

Exemple 5). Les différentes variables listées, identifiées par les balises « variable », apparaissent sous la forme d'un formulaire (cf. Figure 29).

```
<!-- La balise « login » permet de spécifier la présentation de la bannière de
connexion qui va être utilisée pour enregistrer les utilisateurs et recueillir les
informations nécessaires. -->
<login>
        <!-- Les différentes variables listées, identifiées par les balises
         « variable », apparaissent sous la forme d'un formulaire. Le champ « name »
        identifie le nom de la variable qui pourra être évaluée dans le cadre de
        l'affectation dans un groupe d'un utilisateurs. Le champ « style » identifie le
        format des données capturées et indirectement le mode de présentation
        permettant de recueillir l'information (le style integer est matérialisé par un
        objet JSpinner, select par un JComboBox, string par un JTextField, text par un
        JTextArea, etc.) -->
        <variable label="Âge" name="age" style="integer" default="18" />
        <variable label="Statut" name="statut" style="select" default="" />
                <alternative value="Handicapé"/>
                <alternative value="Valide"/>
        </variable>

</login>
```

Exemple 5. Définition des caractéristiques d'une bannière de login

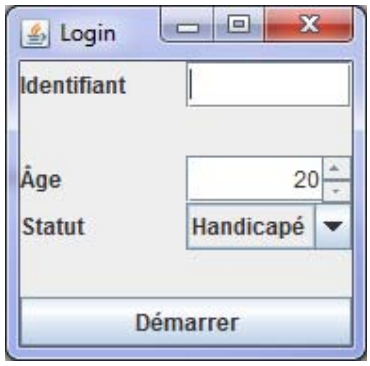


Figure 30. Bannière de login générée à partir de la description XML

```
<!—
     Dans le groupe suivant seront affectés les individus ayant sélectionnés la valeur
     « handicapé » dans la combobox « Handicap » dans le panel de login et dont l'âge
     aura pour valeur entre 10 et 15
-->
<group name="jeunes_handicapés" age="10-15" statut="handicapé" />
```

Exemple 6. Affectation d'un utilisateur à un groupe conditionnée à une valeur saisie dans la bannière de login

Comme nous pouvons régulièrement l'observer [Isokoski 04, Raynal 05d, Merlin 10b], les évaluations expérimentales des dispositifs de saisie sont parfois accompagnées d'un questionnaire recueillant un avis qualitatif sur les différents dispositifs exploités. De manière à gérer le déroulement de ces questionnaires (cf. Exemple 7) au sein même de la plateforme, nous avons additionné un dispositif de questionnaire électronique. Ce dispositif repose sur le même format de spécification que la bannière de login.

```
<questionnary_spec name="QuestionnaireFinal">
        <variable label="Evalués de 1 à 5 les critères suivant :"
                name="" style="label" />
        <variable label="Difficulté de la tâche au moyen du clavier SpreadKey"
                name="spread" style="integer" default="0" />
        <variable label="Difficulté de la tâche au moyen du clavier AZERTY"
                name="spread" style="integer" default="0" />
</questionnary_spec>
```

Exemple 7. Définition d'un questionnaire

Le(s) questionnaire(s) peuvent ensuite être insérés dans les sessions à l'image des exercices ou des instructions (cf. Exemple 8).

```
<session context="gp1:2,4,6;gp2:1,3,5" />
    <instruction texts="instruction-1.txt;instruction-2.html;" />
    <exercise session="1" condition="entrainement" text="mots_0" presentation="mots" />
    <questionnary session="6" name=" QuestionnaireFinal " />
</session>
```

Exemple 8. Mise en place d'un questionnaire au cours d'une session

Quelques protocoles d'expérimentation utilisés au cours de nos travaux sont donnés à titre d'exemple en annexe (cf. Annexe A).[18]

II.2. Accompagnement du déroulement de l'expérimentation

Lors de la première session, l'inscription des utilisateurs est prise en charge par la plate-forme via la bannière de connexion, de login. Les utilisateurs sont automatiquement répartis dans les groupes de manière à équilibrer le nombre d'utilisateurs par groupe, sauf s'il existe des critères spécifiques d'affectation d'un utilisateur à un groupe (comme l'âge ou la condition physique évoqués dans les précédents exemples).

Pour chaque session, la plate-forme va ensuite déterminer le scénario d'expérimentation qui va être suivi en fonction du nombre de sessions déjà effectuées par l'utilisateur et du groupe d'appartenance de celui-ci.

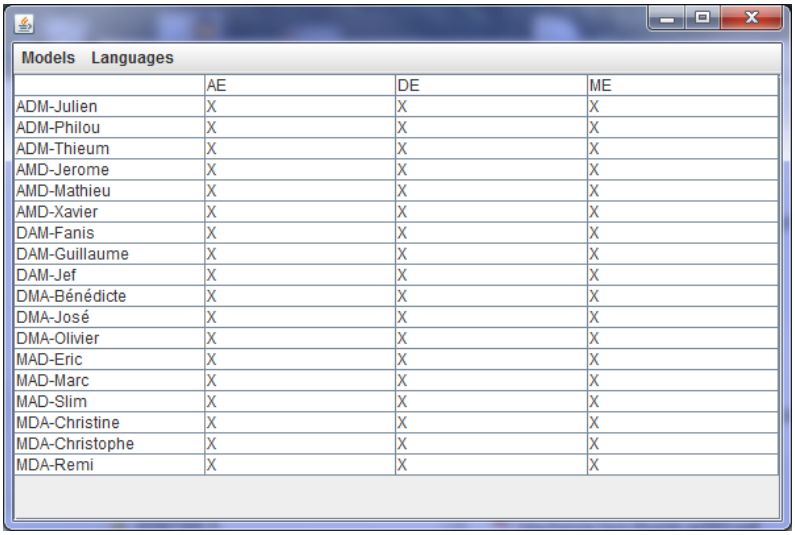
La plate-forme présente ensuite alternativement les bandeaux de consignes et d'exercices en fonction du scénario retenu (cf. Figure 31).

[18] Quelques unes des expérimentations sont par ailleurs accessibles en lignes sur à l'adresse http://www.irit.fr/~Bruno.Merlin/research/index.php?expe=spreadkey

Figure 31. Présentation des consignes (gauche) et exercices (droite)

II.3. Organisation et traitement des résultats

La plate-forme permet ensuite de fournir sous forme graphique (cf. Figure 32) et boîte à moustaches[19] les principaux résultats : distances parcourues par le pointeur (lorsque le pointeur est manipulé via une souris ou un trackball), temps de saisie et erreurs. Les résultats sont présentés par session et/ou par utilisateur et/ou en fonction d'un caractère particulier.



Models Languages

	AE	DE	ME
ADM-Julien	X	X	X
ADM-Philou	X	X	X
ADM-Thieum	X	X	X
AMD-Jerome	X	X	X
AMD-Mathieu	X	X	X
AMD-Xavier	X	X	X
DAM-Fanis	X	X	X
DAM-Guillaume	X	X	X
DAM-Jef	X	X	X
DMA-Bénédicte	X	X	X
DMA-José	X	X	X
DMA-Olivier	X	X	X
MAD-Eric	X	X	X
MAD-Marc	X	X	X
MAD-Slim	X	X	X
MDA-Christine	X	X	X
MDA-Christophe	X	X	X
MDA-Remi	X	X	X

[19] Représentation graphique des valeurs médianes, maximales, minimales et quartiles. Cf. http://fr.wikipedia.org/wiki/Boîte_à_moustaches

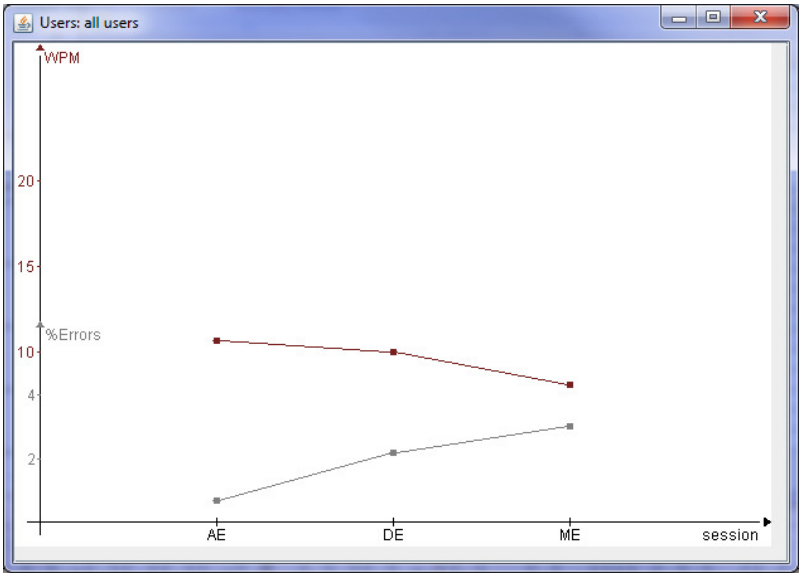


Figure 32. Résultats généraux, par type d'exercice, ou par utilisateur

Par ailleurs, un outil d'analyse (cf. Figure 33) permet d'explorer de façon plus détaillée le déroulement d'une session. Il permet de présenter les résultats caractère par caractère.

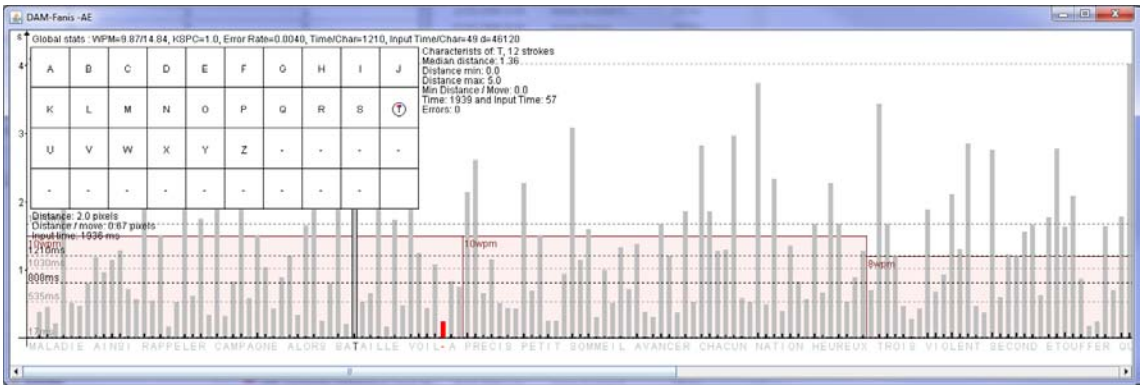


Figure 33. Analyse d'une session

La plate-forme exporte enfin les résultats sous la forme d'un fichier interprétable par les principaux outils d'analyse statistique tels que Satistica[20]. Le fichier fournit l'ensemble des données numériques nécessaires à la réalisation des ANOVA.

Section III - Plate-forme mobile : TinyEAssist

La plate-forme E-Assist II initialement créée permettait de cibler différents contextes d'expérimentation tels qu'un contexte bureautique (expérimentation en local ou via Web), ou qu'un contexte de mobilité au moyen d'un Tablet PC ou une tablette graphique. Néanmoins, les possibilités en termes de mobilité étaient restreintes et réservées aux plateformes accueillant le RunTime Java J2SE (ce qui n'est pas le cas notamment des téléphones mobiles par exemple).

[20] *www.statsoft.com.br/*

En d'autres termes, la plate-forme n'était donc pas exploitable sur les téléphones mobiles ou les Palm qui sont pourtant les principaux dispositifs accueillant des claviers logiciels.

En conséquence, le dernier axe de travail a été de porter la plate-forme E-Assist II en J2ME (Java Mobile) supportée par la très grande majorité des téléphones portables (cf. Figure 34).

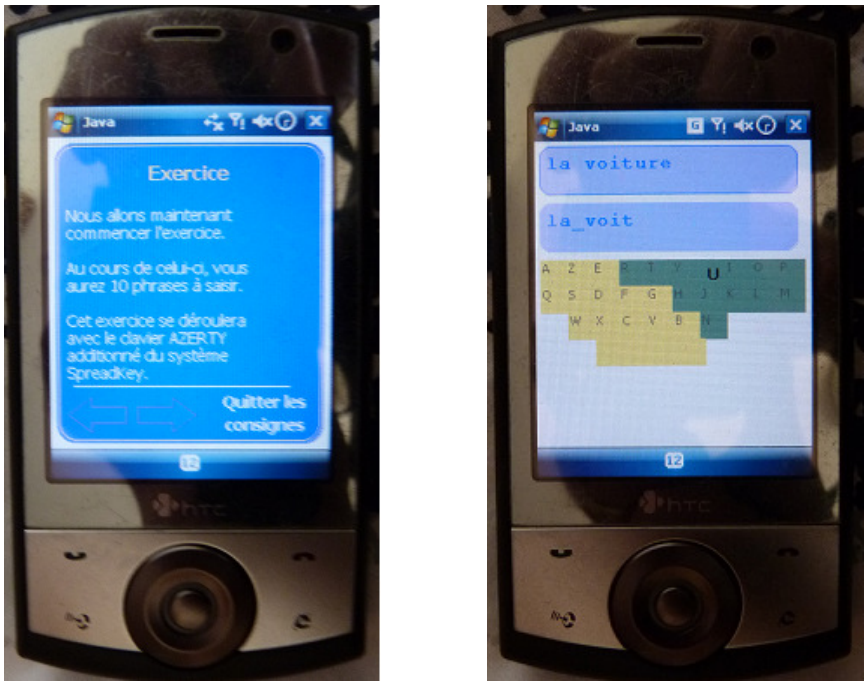


Figure 34. Expérimentation SpreadKey II avec TinyEAssist

Cette version de java offre néanmoins un grand nombre de restrictions en termes de fonctions mathématiques[21] et de fonctions graphiques[22] ayant nécessité le portage parallèle des librairies IntNovate vers intnovateme[23] et Jeks vers jeksme[24] (librairies de rendu graphique et mathématique utilisées par E-Assist II).

La plateforme TinyEAssist (cf. Figure 34) offre ainsi des moyens expérimentaux similaires à EAssist II moyennant quelques restrictions sur les fonctionnalités graphiques[25].

[21] Absence de nombreuses fonctions mathématiques prédéfinies telles que log, acos, pow, exp, etc.

[22] Entre autre, la bibliothèque graphique ne permet pas de dessiner et remplir des formes complexes telles que les courbes de bézier ; elle ne gère pas la transparence ; elle ne gère pas les transformations géométriques telles que scale ou skew.

[23] http://www.irit.fr/~Bruno.Merlin/tools/intnovateme.jar

[24] http://www.irit.fr/~Bruno.Merlin/tools/jeksme.jar

[25] Particulièrement l'absence de la composante alpha dans les couleurs (et donc des effets de transparence) en raison du coût de calcul trop important pour les dispositifs cibles dépourvus de carte graphiques performantes

Section IV - Futurs travaux

Ces premiers travaux nous ont permis de faciliter la conception d'une expérimentation, d'en orchestrer le déroulement et de normaliser le traitement des résultats. Néanmoins, plusieurs initiatives seront approfondies dans le cadre de futurs travaux.

En premier lieu, en matière de conception de l'expérimentation, bien que le support XML soit beaucoup plus simple d'accès qu'un langage de programmation, il reste toutefois encore obscur pour des utilisateurs potentiels tels que des ergonomes où praticiens. Nos futurs travaux nous conduiront donc à proposer une interface graphique de conception des expérimentations permettant une manipulation plus intuitive des fichiers XML.

En second lieu, en matière de traitement des résultats, nos futurs travaux nous amènerons à intégrer le traitement par ANOVA de manière à permettre à la plate-forme de générer un traitement statistique global.

En matière d'accompagnement de l'expérimentation, l'idée future est enfin de fournir, au fur et à mesure de la réalisation des différentes sessions, des indicateurs permettant de vérifier l'évolution de la validité statistique des résultats et permettant de détecter un éventuel problème statistique avant la finalisation même des expérimentations. Ces indicateurs devront notamment nous permettre d'identifier si le nombre d'utilisateurs est suffisant et significatif, de notifier des problèmes dans le contre-balancement des exercices et d'évaluer si l'apprentissage est terminé.

Chapitre 7 - Langage de création des claviers

Si les briques logicielles telles que bandeaux de présentation, présentation des consignes, capture des résultats, etc., peuvent être capitalisées d'une expérimentation à l'autre, en revanche la plupart des solutions évaluées doivent être développées pour l'expérimentation. Le deuxième axe de notre travail sur l'instrumentalisation des expérimentations s'est donc orienté sur la conception des claviers logiciels.

Bien que les claviers logiciels présentent tous des différences, ils manipulent malgré tout de nombreux concepts communs tels que des touches, des paradigmes d'interaction, des systèmes de prédiction, etc. Au cours d'une étude des différents types de claviers logiciels existants (cf. Section I), nous avons identifié ces différents concepts et les avons organisés autour d'une architecture commune (cf. Section II). Nous avons ensuite matérialisé ces concepts sous forme de briques logicielles et proposé un langage XML permettant de générer la plupart des claviers observés (cf. Section III). Nous discuterons des cas particuliers qui échappent aux possibilités initiales offertes par le langage et expliquerons comment le langage peut être étendu pour les intégrer (cf. Section IV).

Section I - Etude des claviers logiciels

Les claviers logiciels sont utilisés dans deux principaux contextes : support d'accessibilité pour des utilisateurs handicapés et moyen principal de saisie sur des dispositifs mobiles. En conséquence, ils sont utilisés sur différents supports (téléphones, écrans tactiles, ordinateurs quelconques) via différents dispositifs d'interaction (souris, trackball, stylet, mouvement de la pupille [Majaranta 07], etc.). Nous avons organisé notre analyse des claviers autour de cinq axes de classification : les claviers logiciels « standards » (reproduction du fonctionnement des claviers physiques sur un support logiciel) ; les claviers ambigus ; les claviers basés sur des styles d'interaction alternatifs ; les claviers logiciels assistés par des systèmes de prédiction et les claviers actionnés par une simple impulsion.

A noter que de nombreux claviers combinent plusieurs stratégies qui s'inscrivent dans différents axes de classification. Il s'agit donc d'analyser et organiser ces stratégies et non les claviers en eux-mêmes.

I.1. Les claviers logiciels standards

A l'exemple du clavier d'accessibilité proposé par Windows, les claviers logiciels standards sont de simples distributions de touches. Les touches sont activées par pression via le dispositif de pointage.

Chaque touche possède au moins deux états graphiques (enfoncé et relâché) et suivant le moyen d'interaction utilisé, un état survolé (notamment lors d'une interaction avec une souris ou un trackball).

Les touches composant les claviers sont de deux natures : soient standards (engendrant la saisie d'un caractère), soient fonctionnelles (leur saisie va modifier le fonctionnement du clavier et altérer les futures frappes).

L'interaction avec une touche standard délivre un évènement au système d'exploitation : une pression sur une touche génère deux évènements que nous dénommerons 'key-pressed' et 'char-input', et le relâchement un évènement 'key-released'. Si la touche est maintenue enfoncée durant un délai T, un nouvel évènement 'char-input' est émis. Si la touche est

maintenue N fois ce délai, l'évènement est émis N fois. Le comportement de ces touches est modélisé par l'automate Figure 35a.

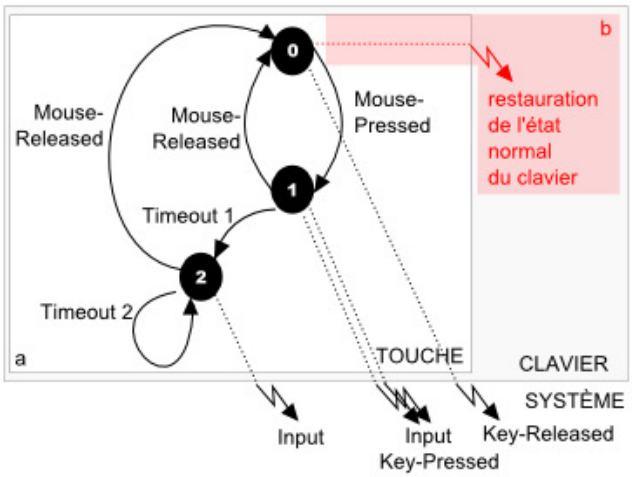


Figure 35. Comportement des touches standards

Les touches de fonction délivrent un évènement au système, mais surtout, elles ont pour finalité d'altérer le comportement interne du clavier. Elles modifient potentiellement les caractères fournis par les autres touches et les messages émis lors des frappes sur ces mêmes touches.

Certaines touches, telles que « Caps-Lock » ou « Num » ont un comportement « permanent ». Plus exactement, leur comportement est effectif jusqu'à ce quelles soient pressées de nouveau (cf. Figure 36a). D'autres, telles que « Shift » ou « Alt » agissent sur des claviers physiques lorsqu'elles sont pressées de façon simultanée avec une touche standard. Or, dans la mesure où la plupart des dispositifs sur lesquels fonctionnent les claviers logiciels, cette simultanéité n'est pas possible. Elle est imitée par la succession d'une pression sur la touche fonctionnelle puis sur une seconde touche. En conséquence, ces touches altèrent uniquement la frappe suivante. Ceci suppose que les touches standards aient également la possibilité d'affecter l'état interne du clavier en restaurant l'état initial (cf. Figure 35b et Figure 36b).

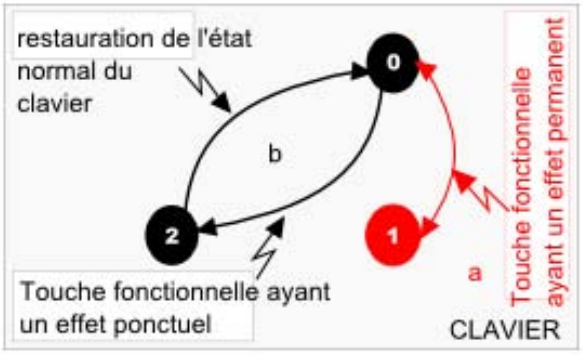


Figure 36. Comportement des touches fonctionnelles

Les travaux d'optimisation proposés autour de ce type de claviers visent à améliorer la compétitivité du clavier en facilitant le pointage. Deux stratégies d'optimisation se distinguent reposant sur les deux variables de la loi de Fitts (W la taille de la touche et D la distance parcourue pour atteindre la cible, cf. Chapitre 1 - I.2) :

- une première stratégie, supportée par de nombreux travaux [Textware 98, Zhai 00, Zhai 02a, Raynal 05b, Bi 10], vise à réduire les distances parcourues entre chaque pointage en réorganisant la distribution de touches. Sur la base des propriétés statistiques du langage ciblé par le clavier, les touches sont réorganisées de manière à rapprocher les lettres dont la co-occurrence est élevée.

- Une seconde stratégie vise à travailler sur la taille des touches de manière à faciliter le pointage. C'est le cas de FloodKey [Aulagner 10] et SmartKey [Al Faraj 09c] augmentant ponctuellement la taille des caractères les plus probables[26], ou encore SpreadKey [Merlin 09a] (que nous aborderons également en troisième partie).

I.2. Les claviers ambigus désambiguïsés par interactions

L'étude des claviers logiciels ambigus (claviers dont les touches regroupent plusieurs caractères) se justifie par la surface réduite des téléphones à écran tactile. Regrouper plusieurs caractères par touches permet ainsi de proposer des touches plus importantes et d'en faciliter le pointage. Ces propriétés intéressent également des projets à destination d'utilisateurs partiellement handicapés des membres supérieurs.

Le concept de clavier ambigu hérite des claviers téléphoniques proposant nécessairement plusieurs caractères par touches en raison du peu d'espace dédié au clavier. Sur des supports à écran tactile, on retrouve ainsi des techniques telles que MultiTap ou Glyph 2 [Poirier 06], conçues initialement pour des claviers téléphoniques. La sélection du caractère est désambiguïsée par plusieurs frappes consécutives. Dans le cas de MultiTap, par exemple, on sélectionne le groupe de caractères et le caractère au sein du groupe de caractères par des pressions répétées sur la touche contenant le caractère (une pression sélectionne le premier caractère de la touche, deux pressions sélectionnent le second, etc.). Il en résulte un nombre moyen de pressions par caractères supérieur à deux. En outre, afin de discriminer N pressions de la touche dans le but de sélectionner le $N^{ième}$ caractère représenté sur la touche ou N pressions sur la touche dans le but de sélectionner consécutivement plusieurs caractères sur la touche, il est nécessaire d'insérer un délai. Glyph 2 propose d'optimiser ce traitement en proposant de presser la touche contenant le caractère désirer afin de sélectionner le groupe de caractère, puis de presser la touche 1 pour saisir le premier caractère du groupe, de presser la touche 2 pour sélectionner le second, etc. Ainsi, le nombre de pression par caractère est réduit à deux, et il n'est plus nécessaire d'introduire de délai pour saisir consécutivement deux caractères d'une même touche.

Pour des claviers logiciels, le support tactile offre par ailleurs des possibilités plus riche en terme d'interaction. Sur la base de distributions différentes des caractères, des claviers tels que MessagEase [Nesbat 03], DashKey [Merlin 09a] ou Claviature[27] offrent une technique d'interaction similaire pour désambiguïser la frappe. Le caractère est sélectionné, parmi le groupe de caractères désigné par pointage, au moyen d'un geste en direction du caractère.

D'autres techniques telles que data-stamp [MacKenzie 02d], Multi-Ring [Raghunath 02] ou encore Tree-based Text Entry [Sandness 04] utilisent deux ou trois touches pour naviguer à travers les distributions de caractères.

Multi-Ring, par exemple, utilise une touche pour sélectionner de façon circulaire un groupe de caractères (une pression sur cette touche permet de sélectionner le groupe de caractères suivant ou de revenir au premier groupe de façon circulaire si le dernier groupe est

[26] Concept de zoom sémantique

[27] http://www.microth.com/claviature/

sélectionné), puis une seconde touche pour sélectionner également de façon circulaire un caractère parmi le groupe de caractères, et enfin une dernière touche pour valider, saisir, le caractère couramment sélectionné.

Tree-based text entry organise les caractères de façon hiérarchique. Les 27 caractères sont en premier lieu divisés en trois groupes de neufs caractères. Trois touches permettent de sélectionner le groupe 1, 2, ou 3. Le groupe sélectionné est lui-même divisé en trois groupes de trois caractères. Les mêmes 3 touches permettent de sélectionner un groupe parmi ces trois qui sera lui-même divisé en 3 groupes ne contenant plus qu'un caractère. Une dernière pression sur l'une des trois touches permet enfin de sélectionner le caractère désiré.

I.3. Désambiguïsation et optimisation par un système de prédiction

De manière à améliorer les performances de la saisie de texte, en réduisant le nombre d'interactions nécessaires à la saisie d'un caracatère (réduction du KSPC), une stratégie consiste à effectuer une désambiguïsation automatique de la frappe sur des claviers ambigus[28]. De nombreux dispositifs d'entrée de textes sont ainsi munis d'un système de prédiction basé sur les statistiques de la langue de l'utilisateur. Ces dispositifs de prédiction peuvent soit interférer au niveau du clavier, soit être intégrés au système de traitement de texte comme c'est le cas par exemple dans VITIPI [Boissière 96, Boissière 00] ou encore de l'auto-correction orthographique dans Word[29]. Dans le cadre de cette étude, vouée au design des claviers logiciels, nous n'étudierons que les systèmes de prédiction interférant au niveau du clavier.

Parmi les systèmes dédiés à la désambigüisation des frappes ambigües on peut bien évidemment citer le T9[30]. De nombreux travaux, tel que les travaux de Levine [Levine 87], de Foulds (clavier TOC) [Foulds 87], de Arnott (clavier Frequency) [Arnott 92], et Lesher [Lesher 98], ont tourné autour de ce concept et ont permis de proposer notamment de nouvelles répartitions des caractères sur les 12 touches de manière à ce que les frappes soient plus efficacement désambiguïsées par ce type d'algorithmes. De nombreux autres chercheurs ont également travaillé sur ce concept en réduisant le nombre de touches à 3 touches ([Sandnes 04]) ou 4 touches (clavier UKO-II [Harbusch 03], [Evreinova 04], [Tanaka-Ishii 02]).

MacKenzie [MacKenzie 01] observe que, si les algorithmes types T9 sont efficaces lorsqu'un mot est présent dans un dictionnaire, en revanche, la saisie, lorsqu'un mot est absent de celui-ci, est extrêmement laborieuse. Pourtant, la saisie de ces mots est extrêmement fréquente étant donné la manière de communiquer via SMS [Grinter 01, Grinter 03]. En réponse, LetterWise [MacKenzie 01] propose une saisie désambiguïsée sur la base d'un trigramme. Le caractère le plus probable présent sur la touche, en fonction des caractères préalablement saisis, est entré sur une simple pression de la touche. En cas d'erreur, l'utilisateur presse une touche « next » (suivant) permettant de saisir le second, puis le troisième caractère le plus probable.

Inspiré par une stratégie équivalente mais profitant d'un clavier accessible via un écran tactile au lieu du clavier physique, Less-Tap [Pavlovych 03] réorganise les caractères sur le clavier en fonction de leur probabilité d'apparition.

De manière à fournir un moyen de corriger les erreurs de prédiction des systèmes de désambiguïsation, ou de manière à fournir un simple outil d'assistance à la saisie [Masui 99,

[28] Sans nécessiter plusieurs frappes par caractère ou tout du moins en minimiser le nombre

[29] *office.microsoft.com*

[30] Reduced keyboard disambiguating system, Patent -US6307549, (1995).

Gong 08], un autre usage fréquent des systèmes de prédiction consiste à fournir une liste de prédiction.

D'autres utilisations plus marginales consistent à : fournir des caractères additionnels sur le clavier, adaptés en fonction des résultats du système de prédiction (comme pour le système KeyGlass [Raynal 05, Raynal 07c]) ; ou à réorganiser le clavier en fonction des données de la prédiction comme pour SpreadKey [Merlin 09, Merlin 10] (cf. également troisième partie) ou Dasher [Ward 00] (cf. I.5).

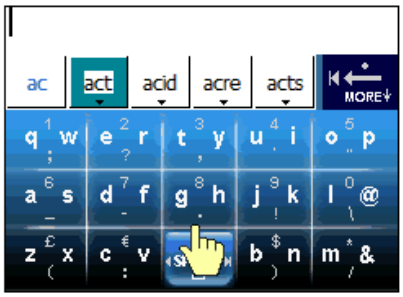


Figure 37. TouchPal

TouchPal[31] (cf. Figure 37) propose un usage combiné de : désambiguïsation par un système de prédiction ; désambiguïsation par interaction ; le tout appuyé par une liste de complétion. L'utilisateur peut choisir entre un geste directionnel désambiguïsant la saisie sur des touches comportant deux caractères ou une simple pression de la touche. Dans le cas d'une simple pression, le plus probable des deux caractères est saisi.

I.4. Les claviers actionnés par une simple impulsion

Dans certains cas de handicap, l'utilisateur n'est pas en mesure de manipuler un dispositif de pointage. Différents dispositifs permettent à cet utilisateur de communiquer avec un dispositif de saisie comme par exemple un système d'acquittement basé sur le mouvement de la pupille [Jacob 93, Isokoski 00, Majaranta 02] ou la détection d'impulsions musculaires [Mason 00, Felzer 05, Felzer 06]. Comme l'utilisateur n'est pas en mesure de pointer une cible (en l'occurrence une touche), le clavier doit automatiquement sélectionner le caractère qui sera saisi. Un mécanisme de défilement permet donc de sélectionner les caractères un à un. Le caractère sélectionné pourra ensuite être validé par une simple impulsion.

Plusieurs stratégies telles que Sibylle [Schadle 04, Wandmacher 07], FLOC [Bellman 98] ou SlideKey [Godard 09], consistent à ré-agencer les caractères en fonction de leur probabilité d'apparition, ou à proposer des stratégies plus complètes pour optimiser la saisie [Wandmacher 08].

I.5. Modalités alternatives d'interaction

Au-delà des dispositifs de reconnaissance d'écriture et de claviers gestuels tel que VirHKey[32] [Martin 05] (cf. Figure 39), EdgeWrite[33] [Wobbrock 03, Martin 08] (cf. Figure 38),

[31] http://www.cootek.com/

[32] VirHKey [Martin 05] propose un alphabet sur la base de deux ou trois gestes directionnels. De manière à guider l'utilisateur dans l'apprentissage de l'alphabet, l'outil propose un feedback visuel.

[33] EdgeWrite [Wobbrock 03] permet une reconnaissance de caractères extrêmement robuste. Un caractère est représenté par un parcours des angles du rectangle. Néanmoins, même si le parcours tend à rappeler la forme du caractère, la saisie n'est pas toujours naturelle pour un débutant. [Martin 08] propose l'addition d'un feedback visuel pour faciliter l'apprentissage.

QuickWriting [Perlin 98] ou Unistroke [Goldberg 97, Isokoski 01], d'autres outils proposent des stratégies de saisie de textes qui dépassent les notions habituelles de claviers.

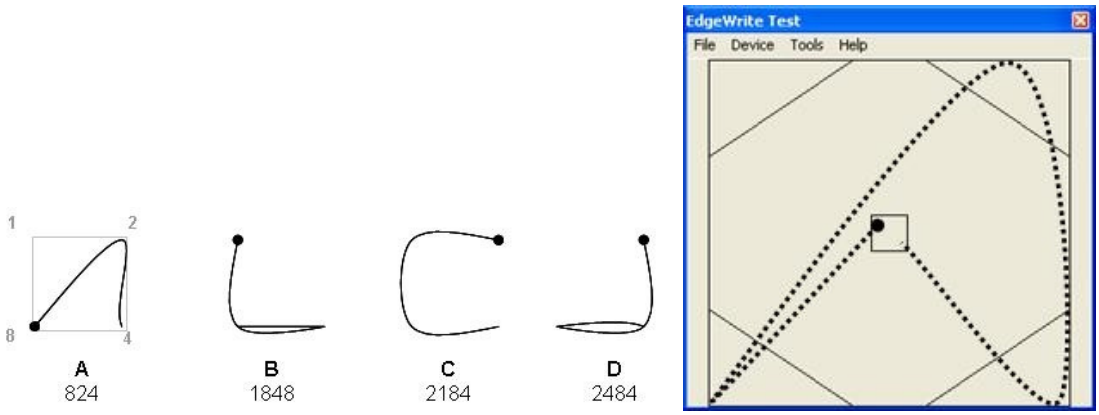


Figure 38. EdgeWrite [Wobbrock 03].

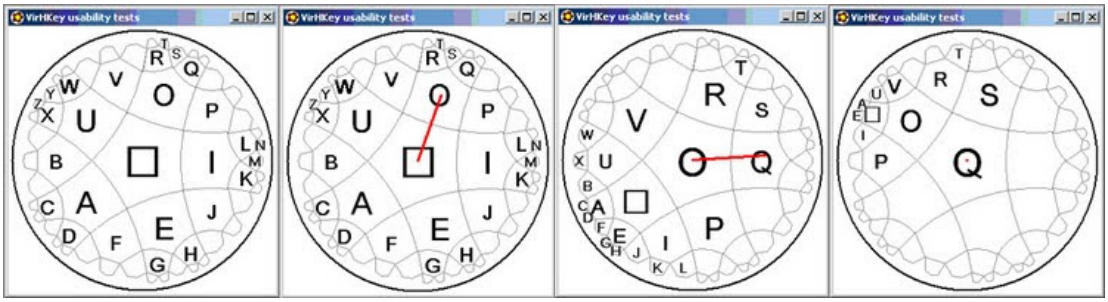


Figure 39. VirHKey [Martin 05]

Par exemple, ATOMIK / ShapeWriter [Zhai 02a, 02b] (cf. Figure 40) permet de saisir les caractères d'un mot sans relever le pointeur. La courbe dessinée sur le clavier est analysée en fonction de la dynamique du tracé et d'un dictionnaire de manière à transformer la courbe en une succession de pointages.

Figure 40. ShapeWriter

Autre exemple, Dasher [Ward 00] (cf. Figure 41) pilote, au moyen du pointeur, le défilement de boîtes contenant chacune un caractère[34]. Ces boîtes ont leur taille proportionnelle à la probabilité d'occurrence du caractère dans le contexte des caractères préalablement saisis. En s'écartant du centre vers la droite, l'utilisateur accélère le défilement des boîtes, en dirigeant le pointeur vers le haut, réciproquement vers le bas, l'utilisateur recentre les boîtes

[34] Cf. la vidéo de présentation de Dasher suivante, http://www.youtube.com/watch?v=0d6yIquOKQ0

du haut, réciproquement du bas. Lorsqu'une boîte traverse le centre du clavier, le caractère qu'elle contient est saisi.

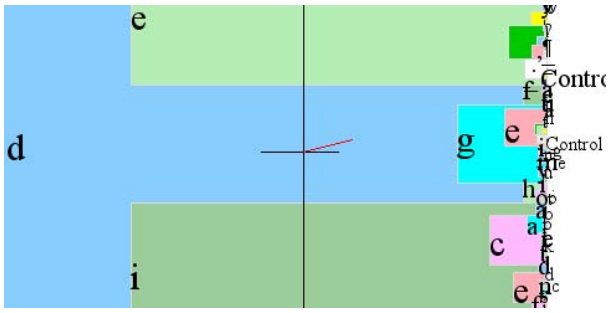


Figure 41. Dasher

Section II - Les entités fonctionnelles des claviers et leurs relations

L'analyse du fonctionnement de ces différents claviers nous a permis de définir cinq entités fonctionnelles communes et constitutives du clavier, ainsi que les relations entre les entités fonctionnelles. La figure 40 illustre les entités, dénommées avec les terminologies suivantes : Interaction Manager (cf. II.1); Layout Manager (cf. II.2); Keys (cf. II.3); Internal State Manager (cf. II.4); Prediction System (cf. II.5), ainsi que leurs relations également exposées dans les différentes sous-sections.

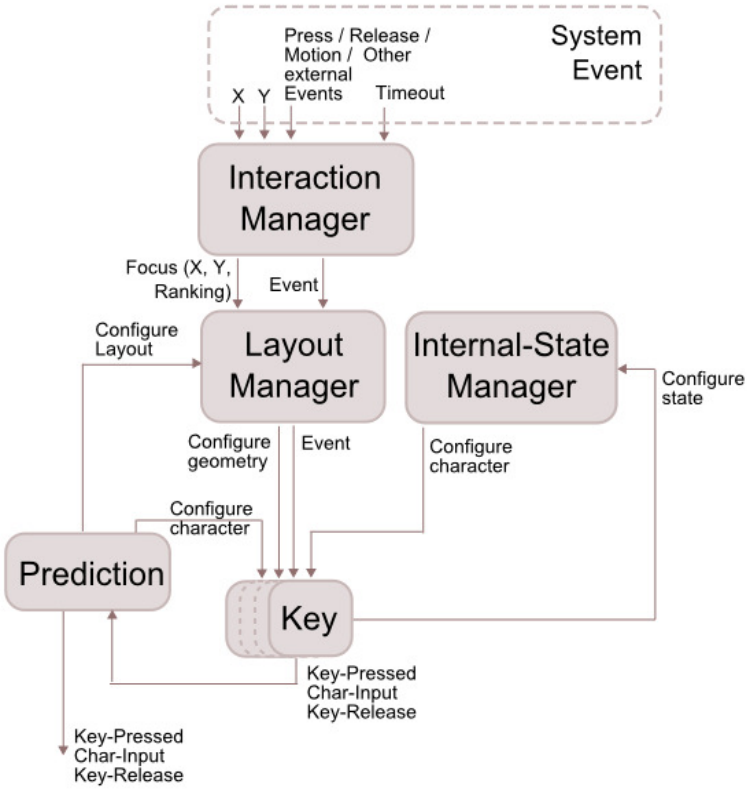


Figure 42. Architecture fonctionnelle des claviers logiciels

II.1. Key (Touche)

L'entité fonctionnelle Key (Touche), gère les différents états graphiques représentatifs de l'état de la touche (par exemple enfoncée, relâchée, survolée, désactivée, shift activé, etc.). Elle organise également l'affichage des caractères, et notifie les évènements clavier (key-pressed, key-released, char-input). A noter, que les évènements claviers ne sont pas directement notifiés au système d'exploitation, mais peuvent être filtrés (voire altérés), par un système de prédiction.

Une touche contient potentiellement plusieurs caractères. La touche va associer la saisie de ces différents caractères à la réception de différents évènements internes. Ces évènements internes sont produits par l'entité *interaction manager* (cf. II.3). Ils sont généralement, mais pas obligatoirement (par exemple à la suite d'un délai), issus de l'interaction avec l'utilisateur.

Chaque touche possède *a minima* deux états graphiques (« relâché » et « enfoncé ») et suivant le moyen d'interaction utilisé, un état « survolé » (notamment lors d'une interaction avec une souris ou un trackball). Pour administrer des claviers tels que Sybille ou FLOC, pour lesquels la sélection d'une touche est automatique et dont l'interaction de l'utilisateur avec le clavier consiste uniquement à valider un caractère sélectionné, il peut être nécessaire de disposer d'un état « sélectionné ».

De manière transversale, les touches fonctionnelles, telles que « Caps lock » ou « Num. », par exemple, peuvent modifier la représentation de la touche dans chacun de ces modes « relâché », « enfoncé », « sélectionné » et « survolé ». Nous avons donc deux notions d'états : une notion d'état de la touche par rapport à l'interaction de l'utilisateur avec le système et une notion d'état du système relatif aux différentes touches fonctionnelles activées ou non.

Le premier état est administré par l'entité Key en fonction des évènements internes reçus. Le second état est général à l'ensemble du clavier et est administré par l'entité *Internal state manager* (cf. II.4).

II.2. Layout Manager

Le *Layout Manager* organise l'agencement et la géométrie des touches. Pour les claviers dont l'agencement est statique, il fait la simple relation entre les coordonnées du clavier et la position des touches. Par ailleurs, dans le cas d'agencements dynamiques, comme pour des claviers tels que SybiMot [Schadle 04], SlideKey [Godard 09], FOCL [Bellman 98], KeyGlass [Raynal 05a], ou SpreadKey [Merlin 09], le *layout manager* gère également le réagencement des touches et caractères.

Il gère également la touche qui détient le focus, par exemple, pour déterminer la touche qui sera sélectionnée dans le cadre d'un clavier contrôlé par impulsions.

II.3. Interaction Manager

A l'image, par exemple, de ce que propose ICON [Dragicevic 02, Dragicevic 04], l'*Interaction Manager* interprète une séquence d'évènements systèmes (principalement les évènements du pointeur et les délais dans le cas des claviers logiciels) et génère un évènement qui sera propagé par le *Layout Manager* jusqu'à l'entité *Key* cible de l'interaction.

Dans le cas de MessagEase, par exemple, l'*Interaction Manager* va devoir transformer une succession d'une pression, de déplacements puis du relâchement du pointeur en un geste directionnel, et discriminer cette séquence d'un simple click. Dans ce cas, un premier

évènement, la pression, permet d'informer le *layout manager* de la touche cible. Un second évènement, après le relâchement, qualifie l'interaction opérée sur cette touche cible.

Autre exemple, dans ShapeWriter, l'*Interaction manager* va scinder une trajectoire continue du dispositif de pointage en une succession de désignations de cibles.

II.4. Internal State Manager

L'entité *Internal State Manager* détermine, en fonction des touches fonctionnelles activées, un état interne du clavier impactant le jeu de caractères actifs. L'état interne peut être modifié par l'appui sur une touche fonctionnelle, mais également par une interaction ou par la pression sur une touche standard, par exemple, après l'appui sur « Shift », l'appui sur une touche quelconque rétablit l'état initial du clavier et impacte donc l'état interne du clavier.

Le changement d'état impacte la distribution des caractères et éventuellement la représentation de la touche.

II.5. Prédiction

Les différents systèmes de prédiction observés reposent sur trois principaux types d'algorithmes : prédiction du caractère suivant en fonction des N précédents caractères saisis et d'un N-Gramme représentatif de la langue considérée [Trnka 06, Wobbrock 06a, MacKenzie 01a] ; prédiction des mots les plus probables en fonction des précédents caractères saisis et d'un dictionnaire (associant ou non la fréquence des mots par rapport à la langue considérée) [Magnien 04, Raynal 05d] ; et, enfin, désambiguïsation d'une séquence de frappes sur des touches ambiguës [US6307549 95] (sur la base également d'un dictionnaire).

L'entité *Prediction* reçoit donc les évènements relatifs à la saisie des caractères émis par les touches (ou groupes de caractères dans le cas de touches ambigües). En fonction de la stratégie de prédiction active, l'entité *Prediction :*

- Propage les évènements aux systèmes d'exploitation et propose des altérations sur le clavier comme par exemple : le réagencement des touches (KeyGlass, Sybille, SpreadKey etc.), ou encore la modification du contenu des touches représentant les items d'une liste de prédiction.

- Mémorise la nouvelle saisie, confronte la séquence saisie avec les caractéristiques statistiques de la langue, et propose une nouvelle séquence destinée à être représentée sur le clavier, ou à destination du système d'exploitation (cas des listes de prédiction, du T9 ou autres algorithmes de désambiguisation, par exemple).

Section III - Langage de modélisation des claviers

De manière à pouvoir implémenter les claviers sur la base de cette architecture commune nous avons proposé un jeu de composants logiciels implémentant ces entités. Chacune de ces entités est largement paramétrable de manière à être adaptable au fonctionnement spécifique de diverses gammes de claviers.

Nous avons par ailleurs proposé un langage XML – KeySpec – permettant de paramétrer et d'assembler ces différentes entités pour ensuite générer un clavier. (La racine XML du langage est la balise *keyboardmodel*.)

Au-delà de l'aspect fonctionnel du clavier, il est également nécessaire de matérialiser son aspect physique. Nous montrerons donc comment nous nous appuyons sur la toolkit IntNovate[35] (II.1) pour définir l'aspect physique du clavier avant d'expliciter le fonctionnement et l'usage des entités *Key* (II.2), *Internal-State-Manager* (II.3), *Layout-Manager* (II.4), *Interaction-Manager* (II.5) et Prediction-System (II.6).

III.1. Gestion de l'aspect physique des touches

L'aspect physique des touches est spécifié en SVG[36] (Scalable Vector Graphics). Les touches peuvent ainsi être définies à partir d'un logiciel de dessin tel qu'InkScape ou Adobe Illustrator. L'utilisation de la toolkit IntNovate, inspirée par les concepts proposés par TkZinc[37], IntuiKit [Chatty 04] et SwingStates [Appert 08], permettra de générer, en Java, les objets dessinés en SVG et de capturer l'interaction effectuée sur les composants graphiques ainsi créés.

Chaque état graphique de la touche (par exemple « relâché », « pressé », « sélectionné », « survolé ») est défini par un nœud de l'arbre SVG, un groupes d'objets graphiques (cf. Figure 43). Le groupe d'objets graphiques correspondant à l'état va permettre de modéliser la forme et l'aspect de la touche, ainsi que la position des caractères. La valeur du caractère sera ensuite spécifiée dans KeySpec (cf. II.2).

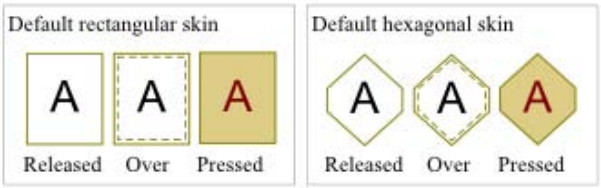


Figure 43. Différents jeux de touches spécifiés en SVG

Une spécification graphique fournit une dimension de référence pour les objets graphiques composant la touche. La spécification de la touche peut cependant être réutilisée pour différentes tailles de celle-ci[38]. Néanmoins, un simple redimensionnement proportionnel de l'image de la touche peut engendrer des déformations de l'image par rapport au sens sémantique des objets (cf. Figure 44).

Inspiré par le concept proposé par Artistic Resizinc [Dragicevic 05], IntNovate permet un re-dimensionnement visant, par exemple, à respecter le sens sémantique des objets graphiques.

La manière de spécifier comment un objet graphique va être redimensionné est fournie par un procédé de design par l'exemple. Au lieu de fournir une représentation unique de la touche, IntNovate offre la possibilité de fournir plusieurs représentations (*keyframes*) ayant

[35]Bibliothèque de design d'interfaces également élaborée en marge des travaux de cette thèse, cf. www.intnovate.org pour plus de détails et exemples de l'ensemble des paradigmes de la toolkit

[36] Le format SVG permet de décrire des images sur la base de formes géométriques. Les entités graphiques (rectangle, cercle, courbe de Bézier, texte, etc.) sont groupées et organisées sous la forme d'une structure arborescente. Les éléments de l'arbre, les nœuds (groupe d'objets) ou les feuilles (objet graphique), peuvent être transformés par application d'une matrice de transformation. Les éléments sont nommés de façon unique.

[37] http://www.tkzinc.org/tkzinc/index.php

[38] Une taille particulière pourra ainsi être sollicitée lors de la spécification du clavier via KeySpec (cf. II.2)

différentes dimensions (cf. Figure 44). Les transformations effectuées lorsque que l'on sollicite une touche d'une dimension quelconque sont obtenues par interpolation entre les deux *keyframes* de dimensions inférieure et supérieure. Chaque élément graphique (dimensions de l'objet, épaisseur du contour, taille des caractères, ancrage des gradients de couleur, etc.) est redimensionné indépendamment par le procédé d'interpolation (cf. Figure 44).

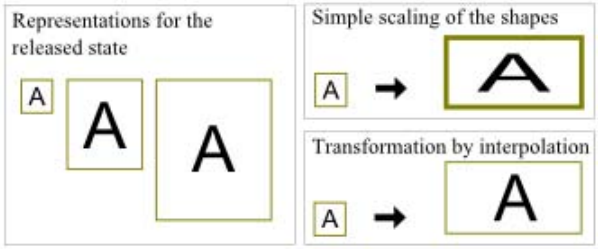


Figure 44. Redimensionnement par interpolation

III.2. Implémentation et exploitation de l'entité *key*

L'entité *key* est manipulée par deux tags du langage KeySpec : *keymodel* et *key*. Comme son nom l'indique, *keymodel* permet de spécifier un modèle, des propriétés communes, pour plusieurs instances d'une touche. Le tag *key* va matérialiser à proprement parler les instances de touches.

Keymodel permet en premier lieu d'associer les différents états graphiques des touches avec des objets SVG (cf. Exemple 9). A noter que KeySpec propose une représentation graphique par défaut dans le cas où les états graphiques ne sont pas explicitement spécifiés.

Key permet ensuite d'instancier un modèle de touche, de configurer la position d'une touche et éventuellement de surcharger des propriétés héritées du modèle.

```
<keymodel name="simple_key"
        released="key.svg#up"
        focused="key.svg#over"
        over="key.svg#over"
        pressed="key.svg#down"
        width="25" height="25" />

<key type="simple_key" x="90" y="30"/>
<key type="simple_key" width="30" height="30" x="120" y="30"/>
```

Exemple 9. Spécification des états graphiques et instances de touches

Une touche peut par ailleurs contenir plusieurs caractères. Ces caractères multiples sont également déclarés lors de la spécification du modèle de touche *keymodel*. Un tag spécifique « *character* » permet d'associer un identifiant au caractère de la touche et une représentation physique à ce caractère. Néanmoins, cette représentation peut être différente pour chacun des états « relâché » (released), « sélectionné » (focused), « pressé » (pressed), « survolé » (over) de la touche.

Le langage permet d'identifier le caractère au sein d'un état graphique par l'intermédiaire d'un suffixe (attribut *svg_suffixe*). Pour identifier le caractère, le suffixe est apposé à l'identifiant du groupe d'objets représentant l'état graphique. Si on reprend l'exemple

précédent, l'état graphique *released* est matérialisé par le groupe d'objets 'up' du fichier 'key.svg' ('key.svg#up'). Un caractère associé au suffixe 'main' sera matérialisé par l'objet 'up_main' dans l'état *released*. De même, dans les états *over* et *focus* le caractère sera représenté par l'objet graphique nommé 'over_main' et dans l'état *pressed* par l'objet graphique nommé 'down_main'.

A noter que, bien qu'en pratique l'élaboration de la structure du fichier SVG ne soit pas complexe, celle-ci exige néanmoins une certaine rigueur et un peu de pratique.

L'exemple suivant (cf. Exemple 10) illustre l'instance d'une touche positionnée aux coordonnées (90, 30) de la fenêtre contenant le clavier. Les dimensions de la touche sont de 30 * 30 pixels et la touche implémente le modèle de touche 'three_character_key'. Les représentations des états graphiques de la touche sont extraites du fichier 'key.svg'. Les états graphiques *released*, *over* et *pressed* sont matérialisés par les groupes d'objet 'up', 'over' et 'down' au sein de ce fichier SVG. Le modèle 'three_character_key' déclare trois objets *character* de noms 'CENTER', 'NO' et 'SE' et associés aux suffixes 'centre', 'NO_corner' et 'SE_corner'. En conséquence, le groupe d'objets 'up' doit contenir trois objets nommés 'up_centre', 'up_NO_corner' et 'up_SE_corner' correspondant à la représentation graphique de ces trois objets pour l'état *released* ; le groupe d'objets 'over' doit contenir trois objets nommés 'over_centre', 'over_NO_corner' et 'over_SE_corner' correspondant à la représentation graphique de ces trois objets pour l'état *over* ; et le groupe d'objets 'down' doit contenir trois objets nommés 'down_centre', 'down_NO_corner' et 'down_SE_corner' correspondant à la représentation graphique de ces trois objets pour l'état *pressed*. L'état *focused* est identique à l'état *up.*

```
<keymodel name="three_characters_key"
        released="key.svg#up"
        over="key.svg#over"
        focused="key.svg#up"
        pressed="key.svg#down"
        width="25" height="25" >

        <character name="CENTER" svg_suffixe="center" />
        <character name="NO" svg_suffixe="NO_corner" />
        <character name="SE" svg_suffixe="SE_corner" />
</keymodel>
<key type="simple_key" x="90" y="30"/>
<key type="simple_key" width="30" height="30" x="120" y="30"/>
```

Exemple 10. Spécification d'une touche contenant trois caractères

Nous discuterons dans la section suivante la valeur des caractères.

III.3. Implémentation et exploitation de l'entité *Internal-State-Manager*

Le comportement de l'entité *Internal-State-Manager* est matérialisé par une machine à états finie (cf. Exemple 11). Les changements d'état sont contrôlés par la notification d'évènements internes émis, soit par une touche, soit par l'entité *interaction-manager*.[39]

[39] Si la machine à états finie se situe dans un état 'X', elle passe dans un état 'Y' sur réception d'un évènement 'EV' si un objet *transition* est présent avec pour paramètre from="X" to="Y" onevent="EV"

```
<fsm default="lowcase">
	<transition from="lowcase" to="upcase" onevent="shift" />
	<transition from="upcase" to="lowcase" onevent="shift" />
	<transition from="upcase" to="lowcase" onevent="standard_key_pressed" />
	<transition from="lowcase" to="maintain_upcase" onevent="caps-lock" />
	<transition from="maintain_upcase" to="lowcase" onevent="caps-lock" />
</fsm>
```

Exemple 11. FSM gérant la casse d'un clavier AZERTY

L'état courant de l'*Internal-State-Manager* permet éventuellement de gérer le jeu de caractères représentés sur les touches (exemple : majuscule ou minuscule) et en conséquence les caractères saisis par une interaction sur la touche, mais il peut également permettre d'afficher ou non un jeu de touches (par exemple, sur le mini-AZERTY, l'affichage du jeu de touches alphabétiques ou du jeu de touches numériques).

En conséquence, KeySpec permet en premier lieu, à travers l'attribut *states* du tag *key*, de restreindre l'affichage d'une touche à certains états de l'*Internal-State-Manager* (cf. Exemple 12). Dans l'exemple suivant, la première touche va être visible lorsque l'état courant de l'*Internal-State-Manager* est 'lowcase' ou 'upcase', la seconde lorsque l'état courant est 'NUM' et la troisième quelque que soit l'état courant (il s'agit également du comportement par défaut lorsque l'attribut *visible_states* n'est pas spécifié).

```
<fsm default="lowcase">
	<transition from="lowcase" to="upcase" onevent="shift" />
	<transition from="upcase" to="lowcase" onevent="shift" />
	<transition from="lowcase" to="NUM" onevent="NUM_event" />
	<transition from="NUM" to="lowcase" onevent="NUM_event" />
</fsm>
<key visible_states="lowcase;upcase" />
<key visible_states="NUM" />
<key visible_states="*" />
```

Exemple 12. Configuration de la visibilité d'une touche en fonction des états de l'Internal-State-Manager

En second lieu, la balise *state* va permettre de spécifier, en fonction des états de l'*Internal-State-Manager*, le(s) caractère(s) représenté(s) sur la touche et les évènements notifiés par une touche lorsqu'une interaction intervient sur la celle-ci (cf. Exemple 13). L'attribut *visible_states* permet de spécifier le nom des états. L'attribut *data* permet de spécifier les données pour la touche dans les états spécifiés par *names*. Ces données sont fournies sous la forme de triplets (caractères représentés, caractères saisis – notifiés au système d'exploitation, évènements internes émis). A noter que « a » est une notation concise pour « (a;a;) » : le caractère représenté est 'a', le caractère notifié au système d'exploitation est 'a' et aucun évènement interne n'est notifié.

Dans l'exemple suivant, lorsque l'*Internal-State-Manager* est dans l'état 'lowcase', le caractère représenté est 'a' ; si l'utilisateur presse la touche, le caractère saisi est 'a' et aucun message interne n'est notifié. Lorsque en revanche l'*Internal-State-Manager* est dans l'état 'upcase', le caractère représenté est 'A'; si l'utilisateur presse la touche, le caractère saisi est 'A' et le message interne 'shift' est notifié à l'*Internal-State-Manager* qui en conséquence passera de l'état 'upcase' à l'état 'lowcase'.

```
<fsm default="lowcase">
        <transition from="lowcase" to="upcase" onevent="shift" />
        <transition from="upcase" to="lowcase" onevent="shift" />
        <transition from="lowcase" to="NUM" onevent="NUM_event" />
        <transition from="NUM" to="lowcase" onevent="NUM_event" />
</fsm>
<key type="default">
        <state visible_states="lowcase" data="a" />
        <state visible_states="upcase" data="(A,A,shift)" />
</key>
```

Exemple 13. Configuration d'un caractère contenu par la touche en fonction de l'état courant de l'Internal-State-Manager

Dans le cas où une touche contient plusieurs caractères, la valeur de l'attribut *data* contient une liste de triplets. Le premier triplet est associé au premier caractère du modèle, le second triplet au deuxième, etc. (cf. Exemple 14).

```
<keymodel name="three_characters_key"
        released="key.svg#up"
        over="key.svg#over"
        pressed="key.svg#down"
        width="25" height="25" >
        <character name="CENTER" svg_suffixe="center" />
        <character name="NO" svg_suffixe="NO_corner" />
        <character name="SE" svg_suffixe="SE_corner" />
</keymodel>
<key type="three_characters_key" width="30" height="30" x="90" y="30">
        <state visible_states="lowcase" data="a;(b,b,);(c,c,input_c)" />
</key>
```

Exemple 14. Configuration de plusieurs caractères contenus par la touche en fonction de l'état courant de l'Internal-State-Manager

III.4. L'entité Layout-Manager

Le *Layout-Manager* gère la géométrie des touches (taille et position), identifie la touche active (détenant le focus) et propage les évènements reçus depuis l'entité *Interaction-Manager* vers la touche active.

La géométrie initiale des touches est définie lors de la déclaration de la touche. L'ordre initial des touches est une conséquence de l'ordre dans lequel celles-ci sont déclarées. Néanmoins un attribut *rank* permet éventuellement de modifier cet ordre (cf. Exemple 15).

```
<key type="…" x="…" y="…"> … </key>    <!-- dimensions déduites du modèle -->
<key type="…" x="…" y="…" height="…" width="…"> … </key>
<key type="…" x="…" y="…" height="…" width="…" rank="0" > … </key>
```

Exemple 15. Modification de l'ordre des touches

Cependant, pour certains claviers, la géométrie des touches particulières peut varier. Dans KeyGlass, par exemple, des touches additionnelles sont positionnées aux angles de la

dernière touche saisie. Autre exemple, POBox [Masui 99] positionne une liste de complétion en fonction de la position de la souris. La première touche, correspondant au premier item de la liste, est positionnée par rapport à la position du curseur tandis que les autres touches sont positionnées par rapport aux précédents items de la liste. La taille d'une touche est déduite de la taille du texte contenu par la touche.

KeySpec permet en conséquence de spécifier la géométrie d'une touche par contrainte[40] en fonction de ces éléments dynamiques. Les variables MOUSE_X et MOUSE_Y matérialisent les coordonnées de la souris, SELECTED_KEY_X / Y / WIDTH / HEIGHT les données géométriques de la touche détenant le focus, PREVIOUS_KEY_X / Y / WIDTH / HEIGHT les données géométriques de la touche précédente dans l'ordre de déclaration des touches et CONTENT_WIDTH / HEIGHT (<identifiant>) les dimensions du texte associé au caractère du *keymodel* de nom <identifiant>. Ces variables peuvent être exploitées pour exprimer les attributs *x*, *y, width*, ou *height* par des expressions mathématiques. L'exemple suivant (cf. Exemple 16), inspiré de la modélisation de KeyGlass (cf. Chapitre 7 - □), positionne la touche dans le coin supérieur gauche de la touche précédemment pressée.

```
<key type="…"
        x="SELECTED_KEY_X – CONTENT_WIDTH(center) / 2"
        y="SELECTED_KEY_Y - 15"
        width="CONTENT_WIDTH(center)">
…
</key>
```

Exemple 16. Positionnement et dimensionnement relatif des touches

A noter que dans des cas plus complexes, comme dans celui de Dasher [Ward 00] ou VirHKey [Martin 05] par exemple, les changements opérés sur les touches dépendent d'algorithmes beaucoup plus complexes qui ne peuvent être matérialisés par de simples relations mathématiques de contraintes. Ces changements devront en conséquence être intégrés par redéfinition de la fonction d'agencement des touches. Ces aspects seront discutés en section IV de ce Chapitre.

III.5. Implémentation et exploitation de l'entité *Interaction-Manager*

L'entité *Interaction-Manager* capture et interprète les actions effectuées par l'utilisateur sur le clavier. Des filtres – des reconnaisseurs –, matérialisés par le tag *interaction*, permettent d'associer l'émission d'un évènement interne avec une ou plusieurs interactions de l'utilisateur.

KeySpec propose un certain nombre de reconnaisseurs prédéfinis (cf. Exemple 17): mouse-pressed (filtre une pression sur dispositif de pointage), mouse-released (filtre le relâchement), mouse-enter (filtre l'entrée du pointeur dans une zone interactive), mouse-leave (filtre la sortie), flick (filtre un mouvement directionnel du pointeur dans une direction encadrée par deux angles), gesture (filtre un geste plus complexe qu'un simple mouvement directionnel sur la base d'un réseau de neurones) et timeout (filtre un délai sans interaction).

```
<interaction type="released" notify="touche_relachee" />
<interaction type="gesture" notify="touche_relachee" gestures="N;UP_ARROW;SOUTH;"/>
<interaction type="flick"
```

[40] Les fonctions mathématiques sont évaluées au moyen de la librairie JeksParser http://www.eteks.com/jeks/

```
                    angle_min="90"
                    angle_min="180"
                    threhold="20" notify="mouvement_SudEst" />
```

Exemple 17. Exemple de déclaration d'interactions prédéfinies

De nouveaux reconnaisseurs peuvent être conçus par composition de reconnaisseurs existants au moyen du tag *interactionmodel*. Par exemple, pour capturer une pression longue sur une touche, nous allons capturer la succession d'une pression sur la touche, d'un délai *t* sans interaction, puis du relâchement de la touche (cf. Exemple 18).

```
<!-- création d'un nouveau modèle d'interaction -->
<interactionmodel type_name="long_press">
        <interaction type="pressed" />
        <interaction type="delay" time="300" />
        <interaction type="released"/>
</interactionmodel>
<!-- instanciation du nouveau modèle -->
<interaction type="long_press" notify="longue_pression" />
```

Exemple 18. Déclaration d'un nouveau filtre d'interaction

D'autres traitements de l'interaction plus complexes et généralement beaucoup plus spécifiques sont parfois requis par certains claviers. C'est par exemple le cas de ShapeWriter [Zhai 03, Zhai 09], qui transforme un tracé continu en une succession de pointages par l'analyse de la dynamique du tracé. Ces traitements ne peuvent être décrits comme une succession d'interactions, cependant, KeySpec offre un mécanisme pour permettre leur intégration qui sera décrit dans la section IV.

Par ailleurs, KeySpec offre également un mécanisme permettant de recevoir le résultat d'un traitement issu d'un dispositif externe (système de reconnaissance vocale par exemple). Les résultats sont reçus via le bus logiciel Ivy [Buisson 02, Merlin 04] sous la forme de messages textuels. Les messages sont filtrés par une expression régulière (cf. Exemple 19).

```
<interaction type="ivy" regexp="RECO_VOCAL sentence=select (.*)" notify="selection" />
```

Exemple 19. Interaction notifiée par un dispositif externe et capturée sur le bus Ivy

L'entité *Interaction-Manager* évalue les reconnaisseurs *interaction* en fonction de leur ordre de déclaration. Lorsqu'un évènement correspondant à une interaction de l'utilisateur est reçu par le clavier, il est mémorisé dans une pile. Chaque reconnaisseur est évalué en fonction de la pile. Si les évènements stockés sont reconnus par un reconnaisseur, l'évènement interne associé au reconnaisseur est notifié, l'évaluation des autres reconnaisseurs est stoppée et la pile d'évènements est vidée.

L'ordre de déclaration des reconnaisseurs est donc prépondérant si nous souhaitons discriminer un click bref d'un click long par exemple. Un click bref correspond à une pression, suivie d'un délai court, suivi d'un relâchement. Un click long correspond à une pression, suivie d'un délai plus long, suivi d'un relâchement. Néanmoins la séquence d'évènements nécessaire pour le click long satisfait également la séquence d'évènements pour le click bref. En conséquence si nous déclarons le reconnaisseur du click bref avant le reconnaisseur du click long, le premier reconnaisseur reconnaîtra systématique la pile d'évènements avant le deuxième reconnaisseur. Le click long ne sera donc jamais reconnu.

Les évènements internes associés aux reconnaisseurs sont notifiés aux entités *internal-state-manager* et *layout-manager.* Un évènement interne issu d'une interaction peut ainsi modifier l'état interne du clavier (ainsi un mode 'numérique' pourrait, par exemple, être activé par reconnaissance vocale, cf. Exemple 20).

```
<interaction type="ivy" regexp="RECO_VOCAL mode numérique" notify="mode_numerique" />
<fsm default="normal">
        <transition from="normal" to="numerique" onevent="mode_numerique" />
        <transition from="numerique" to="normal" onevent="mode_normal" />
</fsm>
```

Exemple 20. Passage en mode numérique effectué à partir d'un système de reconnaissance vocale communiquant les résultats de la reconnaissance sur le bus Ivy

L'évènement notifié au *layout-manager* est propagé à la touche détenant le focus (par défaut la touche couramment pressée ou la dernière touche pressée). La touche peut associer ou non la saisie d'un caractère à la réception de cet évènement. Cette association est créée dans la déclaration du modèle de la touche (cf. Exemple 21). Dans l'exemple suivant, lorsque la touche reçoit un événement 'pression_courte', elle notifie la saisie du caractère 'minuscule' (a/z) et lorsqu'elle reçoit un événement 'pression_longue', elle notifie la saisie du caractère 'majuscule' (A/Z).

```
<interactionmodel type_name="long_press">
        <interaction type="pressed" />
        <interaction type="delay" time="300" />
        <interaction type="released"/>
</interactionmodel>
<interaction type="long_press" notify="pression_longue" />
<interaction type="released" notify="pression_courte" />
<keymodel name="maj_min_key"
        released="key.svg#up"
        over="key.svg#over"
        pressed="key.svg#down"
        width="25" height="25" >
        <character name="minuscule" svg_suffixe="center" on_event="pression_courte" />
        <character  name="majuscule"  svg_suffixe="invisible"  on_event="pression_longue"
/>
</keymodel>
<key type="maj_min_key" x="60" y="30">
        <state names="*" data="a;A" />
</key>
<key type="maj_min_key" x="90" y="30" data="z;Z" />
```

Exemple 21. Saisie des caractères minuscules par un click normal et des majuscules sur une pression prolongée (supérieure à 300ms) de la touche

Dans certains contextes particuliers, la modification de la touche sélectionnée sans l'usage d'un dispositif de pointage est nécessaire. C'est le cas, par exemple, pour l'usage d'un clavier actionné par une simple impulsion tel que Sybilettre [Wandmacher 07, Wandmacher 08]. Un mécanisme, spécialement dédié à cet effet, permet d'altérer le rang de la touche

sélectionnée en spécifiant un incrément pour ce rang ou la sélection d'une touche ayant un rang déterminé. Cette altération du rang est notifiée au *layout-Manager* depuis un filtre d'interaction ou par l'émission d'un évènement interne lors de la pression d'une touche (Cf. Exemple 22). L'exemple suivant illustre qu'au terme d'un délai de 200ms, le rang de la touche sélectionnée est incrémenté ; lorsque la touche 'Maj' (majuscule) est « pressée » sur réception d'un évènement 'STIMULATION' propagé sur le bus Ivy[41] la touche de rang 27 (rang de la touche à partir duquel apparaissent les caractères majuscules) devient la touche sélectionnée ; lorsque une autre touche est « pressée » sur réception de ce même évènement 'STIMULATION', le caractère est saisi et le rang de la touche sélectionnée devient 0.

```
<interaction type="delay" notify ="rank+1" />
<interaction type="ivy" regexp="STIMULATION" notify="pression" />
<keymodel name="touche">
       <character name="char" svg_suffixe="center" on_event="pression" />
</keymodel>
<key type="touche" rank="0" x="0" y="0" data="(Maj,,rank27)" />
<key type="touche" rank="1" x="30" y="0" data="(a,a,rank0)" />
<key type="touche" rank="2" x="60" y="0" data="(b,b,rank0)" />
…
<key type="touche" rank="27" x="0" y="120" data="(A,A,rank0)" />
<key type="touche" rank="28" x="30" y="120" data="(B,B,rank0)" />
```

Exemple 22. Modification de la touche sélectionnée par l'émission d'évènements internes

III.6. Implémentation et exploitation de l'entité *Prediction-System*

L'intégration d'un système de prédiction nécessite de résoudre deux problèmes. En premier lieu, il est nécessaire de spécifier le comportement, l'algorithme et les paramètres linguistiques en entrée du système de prédiction. Et en second lieu, il est éventuellement nécessaire de présenter, sur le clavier, les données fournies dynamiquement par le système de prédiction (au moyen d'une liste de complétion par exemple ou en ré-agençant l'ordre des caractères).

De manière à instancier un système de prédiction, KeySpec offre la balise *prediction*. Trois algorithmes sont implémentés par le langage et peuvent être choisis au moyen de l'attribut *type* : désambigüisation d'une séquence de groupe de caractères (algorithme type T9), prédiction des prochains caractères et prédiction de fins de mots (cf. Exemple 23).

Le système de prédiction est également configuré à partir d'un dictionnaire contenant la fréquence des mots dans le langage[42] considéré. Le dictionnaire est fourni au moyen de l'attribut *dictionary*. La prédiction du caractère suivant peut être établie en fonction du précédent, ou des deux, trois, …, N caractères précédemment saisis. Cette valeur est définie par l'attribut *prefix_size*. Les bigrammes, trigrammes, …, n-grammes sont automatiquement construits à partir du dictionnaire.

```
<prediction name="T9" type="disambiguation" dictionary="francais.txt" />
```

[41] Ceci suppose qu'un dispositif permettant de contrôler les stimulations (par exemple détection d'impulsions musculaires, mouvements oculaires de sélection, etc .) est connecté comme agent sur le bus Ivy. Le dispositif notifie le message « STIMULATION » sur le bus Ivy lorsqu'il détecte une impulsion.

[42] Des dictionnaires sont fournis pour Le français, l'anglais et le portugais-brésilien.

```
<prediction name="bigramme" type="character"
            prefix_size="1"  dictionary="francais.txt" />
<prediction name="trigramme" type="character"
            prefix_size="2"  dictionary="francais.txt" />
<prediction name="completion" type="word" dictionary="francais.txt" />
```

Exemple 23. Instanciation et configuration d'un système de prédiction

Plusieurs systèmes de prédiction peuvent être instanciés simultanément.

Pour la grande majorité des claviers [Tanaka-Ishii 02, MacKenzie 09, Miró-Borrás 09], les résultats issus du système de prédiction sont réutilisés au sein du clavier pour fournir une liste de complétion soit : une succession de touches contenant un ou plusieurs caractères ; dont la position peut être relative ou non à la position du curseur ou de la dernière touche saisie (cf. Section II.4) ; et, dont les contenus sont relatifs aux résultats du système de prédiction.

En conséquence, de manière à rendre le contenu d'une touche relatif au système de prédiction, celui-ci peut être spécifié au moyen des fonctions #PRED(<nom du système de prédiction>,<index du résultat>), #pred, #COMP ou #comp. Les fonctions #PRED et #pred matérialisent les résultats du système de prédiction en majuscule ou minuscule. Les fonctions #COMP et #comp matérialisent le résultat du système de prédiction précédé des caractères du mot déjà saisi. Le caractère de la touche aura automatiquement pour valeur le n-ième (correspondant à <index du resultat>) dernier résultat fourni par le système de prédiction (de nom <nom du système de prédiction>). L'exemple ci-après (cf. Exemple 24) illustre une liste de complétion de trois items positionnée en haut du clavier.

```
<prediction name="completion" type="word" dictionary="francais.txt" />
<interaction type="released" regexp="STIMULATION" notify="pression" />
<keymodel name="touche">
       <character name="char" svg_suffixe="center" on_event="pression" />
</keymodel>
<key type="touche" x="0" y="0" width="100"
       data="(#COMP(completion,1),#PRED(completion,1),)" />
<key type="touche" x="100" y="0" width="100"
       data="(#COMP(completion,2),#PRED(completion,2),)" />
<key type="touche" x="200" y="0" width="100"
       data="(#COMP(completion,3),#PRED(completion,3),)" />
```

Exemple 24. Liste de complétion contenant 3 items

Section IV - Génération des claviers et extension du langage

Une bibliothèque Java permet de compiler et exécuter les claviers décrits au moyen de KeySpec. La génération du clavier fournit un objet « JComponent » contenant le clavier qui peut ainsi être intégré dans une application standard, une application mobile ou web (cf. Figure 45). Par défaut, le conteneur fourni se comporte comme le mini clavier d'accessibilité de Windows : il permet de représenter le clavier dans une fenêtre toujours placée au premier plan propageant les caractères saisis à la fenêtre du système d'exploitation détenant le focus.

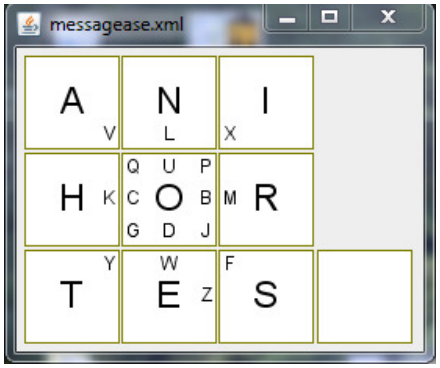


Figure 45. Instance d'un clavier spécifié en KeySpec

IV.1. Génération des claviers

Le processus de compilation transforme chaque objet XML *keyboardmodel*, *keymodel*, *interactionmodel* et *predicitonmodel* sous la forme de classes Java[43].

IV.2. Extension des possibilités du langage

Le comportement particulier des instances des objets *interaction*, *prediction* et *key* est relatif à la valeur passée en paramètre de l'attribut *type*. Ces valeurs correspondent ou bien à des modèles prédéfinis dans la bibliothèque (par exemple les interactions de type 'released', 'pressed' etc., ou les algorithmes de prédiction type 'character', 'word', etc.), ou bien correspondent à des modèles spécifiés par l'utilisateur (par exemple les objets keymodel ou interactionmodel illustrés par Exemple 10 et Exemple 18).

Cette deuxième utilisation est une première forme d'extension du langage. Elle est généralisée (cf. Exemple 25) en permettant le passage, à l'attribut *type,* d'une classe développée par l'utilisateur lui-même[44]. Un attribut *param* permet de passer des paramètres à une classe utilisateur. Dans l'exemple suivant, il s'agit du geste 'up' destiné à être reconnu par le dispositif de reconnaissance de geste users_classes.MyRecoGesture de manière à filtrer l'interaction 'gesture_up'.

```
<prediction name="mine" type="users_classes.UserAlgo" dictionary="francais.txt" />
<interaction name="gesture_up" type="users_classes.MyRecoGesture" param="up" />
```

Exemple 25. Instanciation d'un filtre d'interaction, système de prédiction, etc. à partir d'une classe développée par l'utilisateur

[43] Les classes générées, correspondant aux objets XML *keyboardmodel*, *keymodel*, *interactionmodel* et *predicitonmodel*, étendent respectivement les classes *KeyboardModel*, *KeyModel*, *InteractionModel* et *PredictionModel* fournies par la bibliothèque.

Le clavier généré est donc une instance de l'objet *KeyboardModel*. Les objets XML k*ey, interaction* et *prediction* sont eux-mêmes des instances des classes *KeyModel*, *InteractionModel* et *PredictionModel.* L'entité fonctionnelle *interaction-manager* gère les objets *interaction,* l'entité fonction *layout-manager* organise les instances des objets *key*, l'entité fonction *internal-state-manager* est configurée par l'objet XML *fsm* et l'entité fonctionnelle *prediction* gère les instances des objets *prediction.*

[44] la classe devant étendre la classe *KeyModel*, *InteractionModel* ou *PredictionModel* suivant si elle spécifie le comportement d'un objet *key*, *interaction* ou *prediction.*

Un deuxième procédé d'extension permet de redéfinir, au sein des spécifications de claviers KeySpec, le comportement des fonctions des classes modèles par l'addition de code Java (cf. Exemple 26). L'exemple suivant illustre l'addition d'un feedback sonore lors de la saisie d'un caractère.

```
<keymodel … javaimports="users_classes.*" >
        <function><![CDATA[
                MyAudioFeedback feedback = new MyAudioFeedBack ();
                public void input (String ch) {
                        super.input (ch);
                        feedback.audioFeedback ();
                }
        ]]>
        </function>
</keymodel>
```

Exemple 26. Redéfinition d'une fonction d'un modèle au sein d'une spécification en KeySpec par addition de code Java

Section V - Exemples de claviers

Trois exemples commentés, présents en Annexe B, illustrent les principales possibilités de *keyspec*.

- Le premier exemple[45], DashKey, illustre un clavier contenant des touches ambigües (un caractère central et deux ou trois caractères situés aux coins de la touche). La frappe est désambigüisée par interaction : un simple clic pour la saisie du caractère central, un tracé en direction du caractère pour la saisie des caractères situés aux angles.
- Le second exemple, KeyGlass, illustre un clavier de type AZERTY pour lequel quatre touches additionnelles, contenant les quatre caractères les plus probables, sont proposées aux angles de la dernière touche frappée : le système KeyGlass[46]. Ces touches additionnelles sont donc positionnées de façon dynamique, et leur contenu évolue également de manière dynamique en fonction des résultats d'un système de prédiction. Les touches sont visibles uniquement si le précédent caractère saisi n'est pas un espace. Dans le cas contraire elles sont masquées.
- Le troisième exemple, TouchPal, illustre un autre usage d'un système de prédiction. TouchPal[47] propose deux modes d'utilisation des touches. Les touches ambigües de TouchPal contiennent deux caractères accessibles par un geste (un trait à droite saisit le caractère de droite, un trait à gauche, le caractère de gauche). Une simple pression de la touche saisit le caractère le plus probable des deux caractères présents sur la touche. Ce fonctionnement nécessite la modification du comportement par défaut des touches via l'addition de code Java.

[45] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

[46] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

[47] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

D’autres claviers sont présentés dans l’annexe B ainsi qu’aux adresses :

- http://www.irit.fr/~Bruno.Merlin/research/demo ;
- et, http://www.irit.fr/~Bruno.Merlin/research/demo/spreadkey.html

Section VI - Synthèse

Le langage nous permet de générer la grande majorité des claviers logiciels existants, et, pour une très large proportion, sans nécessité d’insertion de code Java. C’est le cas par exemple de tous les claviers statiques (GAG, Metropolis, Fitaly, AZERTY, etc.), des claviers ambigus désambigüisés par interaction (MultiTap, Glyph2, Multi-Ring, etc.), d’une grande partie des claviers désambigüisés par un système de prédiction (T9, LeterWise, etc. à l’exception pour l’instant de TouchPal), des claviers additionnés de listes de prédiction (claviers standards ou ambigus), et d’une partie des claviers dynamiques (KeyGlass, SibyMot, FLOC, etc.).

Néanmoins, certains claviers évoluant dynamiquement et de façon complexe en fonction des résultats d’un système de prédiction n’échappent pas à des modifications en Java du comportement standard des composants. Ces claviers bénéficient néanmoins de toute la structure de base fournie par le langage et concentre ces intervention programmée sur les détails critiques qui ne peuvent être pris en charge par le langage. C’est le cas par exemple de TouchPal ou SpreadKey.

Enfin, certains claviers dont les transformations graphiques dynamiques sont complexes tels que VirHKey ou Dasher échappent aux capacités du langage.[48]

Le langage a pour avantage de fournir les éléments essentiels et paramétrables pour élaborer et générer des claviers complexes. Il permet en outre d’intégrer automatiquement les claviers dans le système d’exploitation, comme un outil d’assistance standard, ou au sein de la plateforme E-Assist II à des fins d’évaluation, ou encore au sein d’une interface quelconque.

Ainsi, pour créer et adapter l’essentiel des claviers, le langage à pour avantage de ne pas nécessité de connaissances avancées en programmation. Néanmoins, il requiert pour l’instant des notions de XML et éventuellement quelques connaissances sur la structure du format SVG pour qui souhaite élaborer de nouvelles représentations graphiques des touches. Cependant, le caractère descriptif de la conception des claviers via KeySpec permet d’envisager une interface graphique suppléant l’écriture du code XML.

[48] Nous envisageons néanmoins l’intégration d’un modèle de transitions animées qui permettrait, par exemple, la génération de VirHKey

Chapitre 8 - Modélisation et évaluation théorique et heuristique des claviers

Malgré la relativité de la validité des modèles, discutée en première partie, et bien que le poids à attribuer aux différents critères numériques et qualitatifs de comparaison reste à établir, la simulation et l'évaluation théorique restent des outils pour comparer rapidement un grand nombre de solutions différentes sur la base de leurs propriétés mécaniques. Les différentes « solutions » étudiées peuvent concerner de la même manière : les artefacts (des claviers ou des paramètres de claviers) ou l'impact sur les résultats des modèles prédictifs utilisés (autrement dit, l'évaluation des modèles eux-mêmes).

De manière à permettre l'obtention automatique des résultats à partir des modèles prédictifs, mais également de tester différents modèles ou l'impact de certains de leurs paramètres, il est nécessaire de décrire le comportement des claviers face à ces modèles. En conséquence, nous avons étendu KeySpec (cf. Section I) afin de décrire ce comportement et permettre son interprétation par un outil de simulation (cf. Section II).

En Section III, nous décrirons par ailleurs l'outil d'accompagnement des évaluations heuristiques.

Section I - Intégration des modèles prédictifs

KeySpec doit permettre de prendre en compte un modèle mécanique et un modèle cognitif de manière à calculer le temps d'accès au caractère de la touche. Ces modèles peuvent éventuellement dépendre du caractère à saisir. Par exemple dans les claviers MessagEase ou DashKey (cf. Chapitre 3 - II.1), la sélection du caractère central diffère de celle des autres caractères.

La balise *inputmodel* (cf. Exemple 27) permet de décrire un modèle mécanique et un modèle cognitif pour la saisie d'un caractère. Les attributs *fitts_offset* et *fitts_coef* permettent de spécifier les paramètres a et b de la loi de Fitts (Chapitre 1 - I.1**Erreur ! Source du renvoi introuvable.**) et *size* permet de spécifier la taille de la cible (paramètre W) dans le calcul de l'indice de difficulté. Par défaut, les paramètres *fitts_offset* et *fitts_coeff* ont pour valeur 0,083 et 0,127 correspondant aux valeurs expérimentales obtenues par Zhai et al. [Zhai 02b]. L'attribut *size* est par défaut calculé en fonction des dimensions de la touche par l'expression (largeur + longueur) / 2.

L'attribut *cognitive_laps* permet de spécifier le temps de recherche de la cible.

```
        <inputmodel name="caracter" fitts_offset="0"
                fitts_coeff="0.2" cognitive_laps="0" />
```

Exemple 27. Définition d'un inputmodel pour calculer le upper-bound [Soukoreff 95]

Ces différents attributs peuvent être exprimés sous la forme d'une expression mathématique. Dans ces expressions mathématiques, KeySpec permet l'usage de variables préexistantes telles que H (hauteur de la touche) et W (largeur de la touche), ou la déclaration et l'usage de nouvelles variables. Ces variables sont spécifiées à travers la balise *variable*. Leur valeur pourra être modifiée à travers une interface avant une simulation. Ces valeurs peuvent être

booléennes ou flottantes. Leur mode de présentation dans l'interface (checkbox, radio bouton pour les valeurs booléennes, spinner ou simple champ texte pour les autres valeurs numériques) est spécifié par l'attribut *style*. Dans l'exemple suivant (cf. Exemple 28), nous déclarons trois variables nous permettant de calculer les valeurs théoriques en fonction des paramètres proposés par Soukoreff et MacKenzie [Soukoreff 95], Zhai [Zhai 02b] et Accot [Accot 02] pour la loi de Fitts, et deux variables pour spécifier les paramètres de la loi de Hick-Hyman. Les trois premières variables seront présentées dans l'interface de simulation sous la forme de radio-boutons (cf. Figure 46) permettant de mettre à 1 l'une des trois (et à 0 les deux autres). Les deux autres variables apparaîtront sous la forme de *spinner*.

```
<variable name="hick_coeff" style="spinner" default="0.2"
        min="0" min="10" step="0.05" />
<variable name="hick_items" style="spinner" default="1"
        min="1" min="1000" step="1" />
<variable name="mack" style="radio" default="1" group="model" />
<variable name="zhai" style="radio" default="0" group="model" />
<variable name="accot" style="radio" default="0" group="model" />
<inputmodel name="caracter"
        fitts_offset="zhai * 0.083 + accot * 0.145"
        fitts_coeff="mack * 0.2 + zhai * 0.127 + accot * 0.146"
        cognitive_laps="hick_coeff * LOG2 (hick_items)"
        />
```

Exemple 28. Le modèle est défini en fonction de variables déclarées dans KeySpec et éventuellement modifiées avant la simulation

Les modèles sont ensuite associés à des caractères par l'intermédiaire de l'attribut *input_model* de la balise *character*. L'exemple suivant (cf. Exemple 29) illustre l'application de deux modèles différents (simple click, ou franchissement d'une frontière [Accot 02]) différenciant la saisie des caractères centraux et des caractères positionnés dans les angles de DashKey.

```
<inputmodel name="centraux"
        fitts_offset="0.145"
        fitts_coeff=" 0.146"
        />
<inputmodel name="angles"
        fitts_offset="0.165"
        fitts_coeff=" 0.155"
        />
<keymodel name="key">
        <character name="center" svg_suffixe="center" input_model="centraux" />
        <character name="NO" svg_suffixe="NO" input_model="angles" />
        <character name="NE" svg_suffixe="NE" input_model="angles" />
        <character name="SE" svg_suffixe="SE" input_model="angles" />
        <character name="SO" svg_suffixe="SO" input_model="angles" />
</keymodel>
```

Exemple 29. Association d'un modèle à la saisie d'un caractère

Section II - Outils de simulation

Les résultats théoriques sont obtenus au moyen d'un outil de simulation de saisie de textes intégré à la plateforme E-Assist. Pour chaque caractère du texte dont la saisie est simulée, l'outil de simulation calcule le temps nécessaire à celle-ci en fonction : du modèle *inputmodel* associé au caractère ; et de la distance D entre le dernier caractère saisi et la position du nouveau caractère. Le temps considéré est calculé par l'expression suivante.

$$T = fitts\_offset + fitts\_coeff \cdot \log_2\left(\frac{D}{size} + 1\right) + cognitive\_laps$$

Où *fitts_offset*, *fitts_coeff*, *size* et *cognitive_laps* sont les paramètres passés à la balise *inputmodel*. Comme évoqué préalablement, ces mêmes paramètres peuvent être exprimés sous forme d'une expression mathématique. Une interface permet de modifier la valeur des éventuelles variables utilisées par ces expressions (cf. Figure 46) et d'initialiser la simulation.

A noter que nous avons pré-supposé l'usage de la loi de Fitts comme instrument de calcul du temps de pointage. Néanmoins, il est possible d'utiliser un tout autre modèle en spécifiant celui avec l'attribut *fitts_offset* et en affectant 0 à l'attribut *fitts_coeff.*

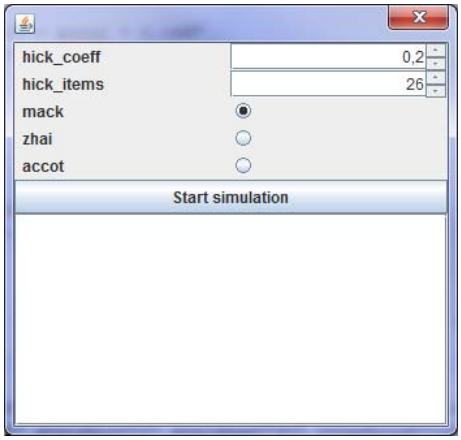


Figure 46. L'interface de simulation permet de modifier la valeur des variables déclarées dans le fichier KeySpec

A noter que, pour que les résultats théoriques soient représentatifs, il est nécessaire que les textes utilisés soient également représentatifs de la langue considérée. En conséquence, l'outil de simulation propose deux possibilités : ou bien la simulation de la saisie des mots d'un dictionnaire pondérée par la fréquence des mots dans la langue considérée, ou bien la saisie de textes.

Il propose également un instrument de préparation de textes. Cet instrument permet de confronter un texte aux données statistiques de la langue : fréquence des caractères, fréquence des bigrammes et fréquence des mots. Ces données statistiques sont automatiquement déduites des dictionnaires contenant la fréquence des mots et utilisés par les systèmes de prédiction évoqués dans KeySpec. L'instrument indique en pourcentage le degré de corrélation du texte avec ces données statistiques.

Section III - Outils d'évaluation heuristique

L'outil d'évaluation heuristique est conceptualisé comme un questionnaire à destination de l'expérimentateur. Afin de pouvoir adapter de manière flexible les critères évalués au cours de nos futures recherches, le questionnaire est spécifié en respectant le format proposé au Chapitre 6 - II.1.

Néanmoins, pour tenir compte de la cohérence du format multi-colonnes du questionnaire, la balise de formatage *layout* a été additionnée (cf. Exemple 30).

```
<layout column="5" name="Critères relatifs à l'objectif de l'utilisateur"  />
        <variable label="" style="label" />
        <variable label="SMS Informel" style="label" />
        <variable label="SMS Formel" style="label" />
        <variable label="Email" style="label" />
        <variable label="Documents formatés" style="label" />

        <variable label="Facilité d'accès aux caractères accentués" style="label" />
        <variable label="" name="critere1_1" style="integer" default="0" />
        <variable label="" name="critere1_2" style="integer" default="0" />
        <variable label="" name="critere1_3" style="integer" default="0" />
        <variable label="" name="critere1_4" style="integer" default="0" />
<layout/>
```

Exemple 30. Layout de la grille d'évaluation heuristique

L'outil effectue par ailleurs le traitement des résultats une fois l'ensemble des critères saisis pour le clavier (la somme des résultats pour chaque colonne et pour chaque groupe de colonnes regroupées au sein d'un objet *layout*). La balise permet également d'obtenir une note globale en fonction d'une combinatoire de paramètres en permettant de sélectionner une colonne (par son nom) pour chaque objet *layout*.

Conclusion

Les travaux réalisés autour de la plateforme E-Assist II nous ont donc permis d'élaborer un outil complet d'étude des claviers logiciels. Outre le langage de spécification des claviers KeySpec permettant de faciliter la conception des claviers, E-Assist II propose désormais un ensemble d'outils permettant de comparer les claviers sur les bases d'une variété de données théoriques, ergonomiques et expérimentales envisagées et préconisées dans le cadre de cette étude.

L'essentiel des travaux évoqués en partie I et III de ce manuscrit ont été réalisés avec le support de E-Assist II (dans ses versions finales et intermédiaires). La flexibilité de l'outil nous a permis, entre autre, d'effectuer des démarches expérimentales dans des contextes aussi divers que les environnements bureautiques (en local ou via web), et mobiles (sur Tablet PC ou téléphones portables).

Enfin, avec la perspective finale de permettre l'utilisation complète de la plateforme par un public plus large que la communauté des chercheurs en interaction Homme-Machine, nous avons initié des travaux [HA 10] visant à créer un outil graphique accompagnant et guidant la conception des claviers logiciels. Ces travaux ont pour première finalité de permettre à des thérapeutes accompagnant des enfants handicapés mentaux et moteurs, de concevoir des claviers logiciels adaptés sur mesure aux besoins de leur patient.

Troisième Partie

Nouveaux paradigmes : claviers quasi AZERTY

Comme nous l'avons suggéré au cours de ces deux premières parties, si nous pouvons assez facilement proposer un clavier logiciel aux performances maximales meilleures que celles du clavier AZERTY (ne serait-ce qu'un clavier « ZAERTY »), il est revanche moins évident d'en faire bénéficier collectivement une population. L'acceptation d'un nouvel outil a bien évidemment des origines sociologiques et commerciales que nous ne saurons mesurer dans cette étude. Néanmoins, nous défendons le fait que les propriétés techniques intrinsèques d'un outil, mises en évidence par les évaluations heuristiques, peuvent favoriser (ou non) son acceptation. Parmi ces propriétés, les performances maximales accessibles avec l'outil n'ont qu'un impact limité, à la différence, par exemple, de la non régression des performances au cours de l'apprentissage, et l'absence d'un coût cognitif élevé relatif à l'apprentissage ou à l'utilisation du clavier.

Cette partie aborde ainsi les conséquences de cette réflexion sur nos axes de recherche et les différentes pistes que nous avons étudiées, ciblées autour de deux démarches : le maintien des références de l'utilisateur ou l'accompagnement de l'utilisateur dans la transition. Ces caractéristiques sont privilégiées et ce au détriment de la compétition pour obtenir de meilleurs performances maximales. Cette réflexion nous a amené à étudier, en premier lieu, trois axes de recherche autour du concept d'optimisation « *quasi-QWERTY* » [Bi 10].

Dans un premier temps, nous résumerons (Chapitre 12) les principales solutions traditionnellement développées pour optimiser la saisie.

Puis, dans un second temps, nous présenterons l'expérience acquise autour du clavier SpreadKey (Chapitre 13) dont la finalité est d'atteindre l'objectif, a priori paradoxal, de réduire drastiquement les distances parcourues par le dispositif de pointage et d'augmenter de façon importante la taille des touches, tout en conservant pour base un clavier AZERTY.

Nous présenterons ensuite les travaux réalisés autour des claviers à pointage sémantique, SemanticKeyboard et SemanticKeyboard Mobile (Chapitre 14) qui poursuivent un objectif similaire mais en conservant strictement l'apparence d'un clavier AZERTY. Les claviers à pointage sémantique modifient l'espace physique (la dynamique du déplacement du curseur ou bien la taille de la cible en fonction de la probabilité de saisie des caractères) sans modifier l'espace visuel.

Enfin, nous présenterons nos travaux menés et à venir autour des concepts de clavier multi-layer (Chapitre 15) visant à accompagner progressivement l'utilisateur vers l'usage d'une nouvelle disposition des caractères ainsi que (Chapitre 16) les perspectives autour des

claviers complets (ne comportant pas uniquement 27 caractères comme les claviers souvent comparés dans les études).

Chapitre 9 - Optimiser la saisie

Les travaux publiés dans la littérature visant à l'amélioration de la saisie abordent essentiellement deux axes : l'optimisation du pointage des caractères en travaillant autour des caractéristiques de la loi de Fitts (I) ; et la réduction du nombre de pointages en proposant notamment des stratégies de complétion (II).

Section I - Optimisation du pointage

Considérant immuables les caractéristiques psychomotrices d'une population, et considérant un dispositif d'interaction donné, la loi de Fitts contient deux variables pouvant influer sur les performances : la distance parcourue par le pointeur entre les pointages ; et la complexité de la tâche matérialisée par la dimension des touches.

Plusieurs travaux préalablement explicités [Lesher 00, Zhai 00, Zhai 02a, Raynal 05b] visent à réduire la distance parcourue par le pointeur en réorganisant la distribution des touches sur le clavier et en rapprochant les caractères dont la co-occurrence est fréquente (ces réorganisations tiennent donc compte de la langue de l'utilisateur). Néanmoins, comme nous l'avons déjà évoqué, face à ces nouveaux claviers, un utilisateur déjà aguerri au clavier AZERTY retrouve le statut de débutant avant de progresser vers des performances meilleures [MacKenzie 99]. La phase nécessaire à l'apprentissage des claviers est très préjudiciable à leur acceptation.

De façon alternative, Isokoski [Isokoski 04] propose de compléter le pointage d'une touche par la saisie d'un second caractère au moyen d'un marking menu (cf. Chapitre 2 - Section I -). Néanmoins, cette technique reste relativement complexe cognitivement. Raynal [Raynal 05a, Raynal 05d] propose également quatre touches additionnelles (cf. Chapitre 2 - I.1), positionnées aux angles de la dernière touche saisie, contenant les caractères les plus probables en fonction des caractères préalablement saisis. Mais, dans la mesure où ces caractères sont adaptés dynamiquement, l'utilisateur ne peut généralement pas anticiper leur contenu et le temps nécessaire à la lecture de ces touches additionnelles ne compense pas le temps gagné sur le pointage. Le clavier reste néanmoins efficace pour des utilisateurs handicapés moteurs car il permet de réduire les distances parcourues par le dispositif de pointage et ainsi la fatigue engendrée par sa manipulation.

L'augmentation radicale de la taille des touches, sans modifier les dimensions du clavier, est naturellement confrontée à des limites géométriques. En dehors de certaines stratégies consistant à mieux profiter de l'espace en proposant, par exemple, des touches hexagonales et [Al Faraj 09a, Zhai 00], d'autres techniques consistent à fournir un zoom « linguistique » [Al Faraj 09b, Al Faraj 09c, Aulagner 10] sur les touches dont les caractères sont plus probables, ou bien un fisheye permettant d'augmenter les touches de la zone où se situe le pointeur [Raynal 07a, Raynal 07b]. Néanmoins, la principale solution exploitée pour augmenter la taille des touches reste de grouper les caractères sur des touches ambigües et de désambigüiser la frappe au moyen d'une interaction [Nesbat 03, Merlin 10b] ou d'un système de prédiction [US6307549 95, Tanaka-Ishii 02, Harbusch 03] voire des deux (cf. TouchPal, Chapitre 2 -).

Section II - Réduction du nombre de pointages

Une stratégie fréquente pour permettre la réduction du nombre de pointages est la fourniture de listes de complétion [Masui 99, Tanaka-ishii 07, Badr 09, Trnka 09]. Néanmoins, dans la plupart des cas, l'utilisateur n'est pas en mesure de prédire le résultat de la liste de prédiction. En conséquence, ces listes de complétion nécessitent une activité de lecture parallèle à la tâche de saisie qui ne les rend pas toujours très efficaces [James 01].

En dehors de la technique proposée par Isokoski (cf. Section précédente) consistant à compléter le pointage par un marking menu, d'autres stratégies s'adressent plus spécifiquement aux claviers ambigus.

Les techniques d'interaction sur claviers ambigus (Multitap, Glyph 2 [Poirier 06], Wristwatch [Raghunath 02], Multi-Ring [Sandnes 04]) requièrent plusieurs interactions pour la saisie d'un unique caractère. Les supports tactiles ont permis des modes de désambigüisation alternatifs basés sur une interaction gestuelle comme par exemple MessagEase [Nesbat 03], ou DashKey [Merlin 10b].

Néanmoins, sur les claviers physiques, une manière efficace de réduire le nombre de frappes nécessaires à la saisie est de proposer une désambigüisation automatique de la frappe, basée sur un système de prédiction dépendant des caractéristiques linguistiques de la langue de l'utilisateur (existence et fréquence des mots / trigramme / bigramme etc. dans la langue considérée).

Le système de désambigüisation le plus répandu est le T9. Cependant, plusieurs travaux ont mis en évidence que la distribution alphabétique des caractères sur les claviers téléphoniques est sous-optimale par rapport à la désambigüisation. Ils ont, en conséquence, proposé d'autres distributions des caractères pour améliorer le rendement des systèmes de prédiction [Foulds 87, Arnott 92, Levine 87, Lesher 98]. Ces stratégies de réorganisation des caractères sont néanmoins pénalisées car elles contraignent l'utilisateur à un apprentissage (par rapport à l'ordonnancement alphabétique traditionnel).

MacKenzie [MacKenzie 01a] observe que le système T9 est peu efficient lorsqu'un mot n'est pas présent dans le dictionnaire du système. En outre, il nécessite de nombreuses ressources (notamment de mémoire pour stocker et charger le dictionnaire). En réponse, il propose LetterWise dont le système de prédiction est basé sur un trigramme. Le caractère le plus probable de la touche ambigüe pointée est saisi. Si le caractère ne correspond pas au caractère désiré par l'utilisateur, l'utilisateur presse une touche 'Next' qui remplace le caractère saisi par le second caractère le plus probable, puis respectivement par le troisième caractère de la touche. Sur un clavier manipulé à travers un écran tactile, LessTap s'inspire de cette technique et propose un feedback visuel sur l'ordre des caractères de manière à faciliter l'anticipation par l'utilisateur du nombre de fois où il sera nécessaire de presser la touche « Next ». Ces différentes techniques nécessitent néanmoins un nombre moyen de frappes par caractère supérieur à un.

Chapitre 10 - SpreadKey

L'initiative relative au clavier SpreadKey repose sur l'observation du fait qu'accroître la taille des touches tout en réduisant les distances parcourues par le pointeur serait une solution efficace pour améliorer les performances du clavier. Néanmoins les propositions existantes pour atteindre l'un ou l'autre de ces objectifs :

- Sont soit difficilement acceptables par l'utilisateur (comme le cas de la réorganisation des caractères) ;
- Soit n'ont pas un apport très significatif sur le pointage;
- Soit augmentent le nombre de frappes nécessaires pour la saisie d'un caractère ;

Par ailleurs, les apports dynamiques tels que les KeyGlass ou les listes de complétion, ainsi, *a fortiori,* que les réorganisations dynamiques de la distribution des caractères [Bellman 98, Schadle 04], ne permettent pas à l'utilisateur d'anticiper leur usage introduisant un coût cognitif souvent supérieur au gain de temps qu'ils apportent pour des utilisateurs aux capacités motrices normales.

L'objectif de SpreadKey est donc de résoudre le paradoxe d'augmenter la taille des touches et de réduire les distances tout en maintenant les caractères accessibles à leur place originelle (avec pour finalité première de permettre l'anticipation du pointage du caractère suivant par les utilisateurs).

Section I - Concepts

Le clavier SpreadKey[49] est basé sur un clavier standard AZERTY (ou QWERTY / QWERTZ etc.) et utilise un système de prédiction classant les caractères par probabilité d'occurrence, après chaque caractère saisi, et en fonction de tous les caractères du mot préalablement saisis.

L'idée de SpreadKey est de substituer temporairement les caractères ayant une probabilité nulle ou quasi-nulle d'être saisis par un caractère ayant une probabilité élevée. Autrement dit, en fonction des caractères préalablement saisis, un caractère ***privilégié,*** ayant une probabilité élevée, va devenir accessible, par un simple clic sur les touches ***recyclées*** contenant originellement un caractère de probabilité nulle (ou considérée comme telle) dans le contexte de la saisie. Ceci de façon additionnelle car le caractère ***privilégié*** reste également accessible sur sa touche normale.

Néanmoins, en cas de saisie d'un mot absent du dictionnaire (nom propre, sigle, mot étranger, vocabulaire spécialisé, etc.), un caractère requis par l'utilisateur risque d'être recyclé par le système. Pour permettre la saisie de ces caractères, un caractère ainsi recyclé reste accessible à travers une interaction alternative au simple clic (pression longue ou interaction gestuelle simple). Le caractère originel de la touche reste visible dans l'angle supérieur gauche de la touche (cf. Figure 47).

[49] Démonstration accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo/spreadkey.html

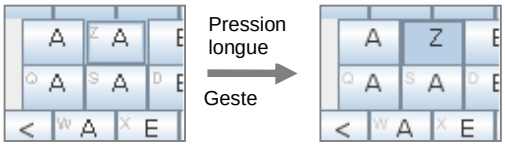


Figure 47. En cas de pression longue ou de geste sur une touche recyclée, le caractère originel de la touche est saisi

Les touches de SpreadKey sont donc des touches ambigües mais dont la désambigüisation fonctionne dans 100% des cas pour la saisie d'un mot présent dans le dictionnaire.

Par ailleurs, la stratégie de recyclage des touches a pour objectif de :

- Réduire les distances en insérant des instances des caractères les plus probables entre la dernière touche pointée et la touche contenant originellement le caractère (cf. Figure 48). Néanmoins, dans la mesure où les caractères restent également accessibles à leur place originelle, le clavier pourrait être utilisé comme un simple clavier AZERTY. Ainsi, l'utilisateur doit être en mesure d'initier le pointage avant d'analyser l'ensemble des modifications issues du système de prédiction.

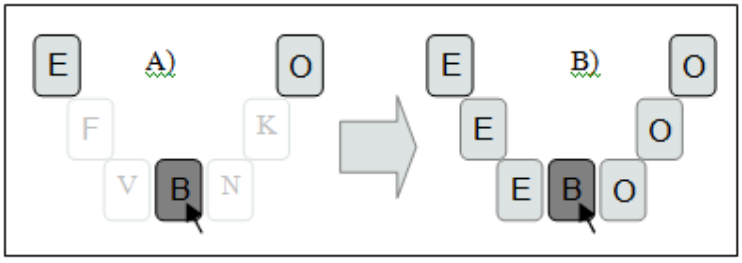


Figure 48. Réduire les distances

- Augmenter la taille des touches en recyclant les touches voisines d'une touche contenant un caractère fortement probable avec des instances de ce caractère (cf. Figure 49).

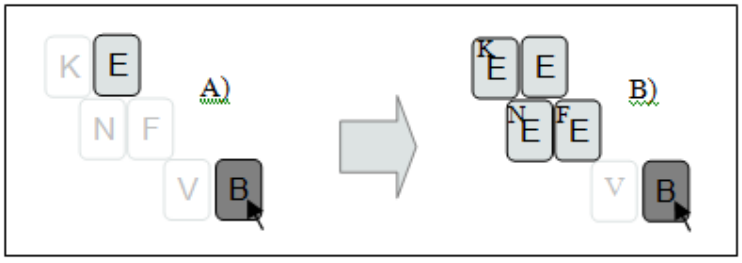


Figure 49. Augmenter la taille des touches

Dans la première version de SpreadKey, l'algorithme de recyclage des caractères tend à substituer les caractères en fonction :

- De la distance de la touche recyclée par rapport à la touche contenant normalement le caractère ;

- De la probabilité d'occurrence du caractère.

Par ailleurs, deux paramètres vont permettre d'influer sur les performances de l'algorithme de recyclage :

- La probabilité à partir de laquelle nous considérons un caractère comme un caractère privilégié (le paramètre, exprimé en pourcentage, sera noté **P**). Si la probabilité d'occurrence du caractère est supérieure à P, le caractère pourra être utilisé pour recycler des touches. Si la valeur de P est élevée, le nombre de caractères *privilégiés* sera plus faible et réciproquement, si P est faible, le nombre de caractères *privilégiés* sera élevé.

- La probabilité à partir de laquelle nous considérons un caractère comme un caractère recyclé (le paramètre, exprimé en pourcentage, sera noté **R**). Pour pouvoir entrer tous les mots du dictionnaire en ne procédant que par de simples clics, cette probabilité doit être égale à 0. Néanmoins, de manière à augmenter le nombre de touches recyclées, il est possible de négliger les cas où la probabilité d'un caractère est très faible. Nous considèrerons donc comme improbables et recyclables les caractères dont la probabilité est inférieure au paramètre R.[50]

L'impact des paramètres et leur valeur seront discutés dans la section suivante.

Plusieurs caractères *privilégiés* sont ainsi en concurrence pour recycler une même touche. La fonction mathématique définissant les caractères de substitution est la suivante :

$$\forall i \quad Pi \geq P, \forall j \quad Pj \leq R$$

$$C_j \textit{ prends la valeur de } C_k \quad \Leftrightarrow \quad f(C_i, C_k) = \max(f(C_i))$$

$$\textit{avec} \quad f(C_i) = \frac{\sqrt{P_i}}{d_i}$$

Où :

P_i est la probabilité d'occurrence du caractère C_i, P et R sont les paramètres explicités ci-dessus, d_i est la distance entre un caractère et la touche contenant normalement le caractère.

[50] A noter que lorsque nous augmentons R, nous augmentons le nombre de cas où l'utilisateur devra recourir à une interaction plus complexe (pression longue ou geste) pour saisir un caractère. Néanmoins cette solution reste plus intéressante que de retirer les mots peu fréquents du dictionnaire.

En effet, supposons une séquence de caractères rare dans la langue française : « ist » requis pour la saisie du mot « isthme ». Lorsque nous saisissons « is », la probabilité de saisir un « t » dans la langue française est très faible dans la mesure où seul le mot « isthme » commence par la séquence « ist ». Nous pouvons donc négliger ce cas et considérer le « t » comme improbable. Dans la situation exceptionnelle où l'utilisateur souhaite réellement saisir le mot « isthme », il devra employer une interaction plus complexe pour saisir le caractère « t ». Mais après la saisie de la séquence « ist » puis « isth » et « isthm » la probabilité de saisir un « h » puis un « m » et un « e » est de 100% et le mécanisme de SpreadKey sera donc d'une efficacité maximale en recyclant l'ensemble du clavier avec le caractère désiré.

Si nous retirons le mot isthme de notre dictionnaire, après la saisie de la séquence « is », l'utilisateur devra utiliser une interaction complexe pour saisir le caractère « t ». Ensuite tous les caractères auront une probabilité nulle d'être saisis. Le clavier fonctionnera alors comme un clavier AZERTY simple pour la saisie de la fin du mot sans bénéficier de l'optimisation de SpreadKey.

A noter que dans cette opération, toutes les touches dont la probabilité d'occurrence est comprise entre R et P (caractères ni privilégiés, ni recyclables) restent inchangées.

Idéalement SpreadKey devrait permettre de créer des « bulles de bande dessinée » (cf. Figure 50), pointant sur le caractère préalablement saisi et contenant les caractères les plus probables dans le contexte de la saisie.

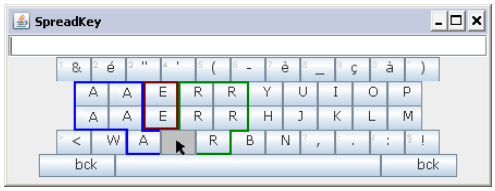


Figure 50. Recyclage idéal des caractères

Dans la réalité, l'algorithme donne lieu à des situations très hétérogènes en fonction de la nature du préfixe déjà saisi. D'une manière générale, plus le nombre de caractères du mot déjà saisi est élevé, plus le déterminisme lié au caractère suivant est élevé (cf. Figure 51a, b et c). En conséquence, SpreadKey recyclera un maximum de touches du clavier.

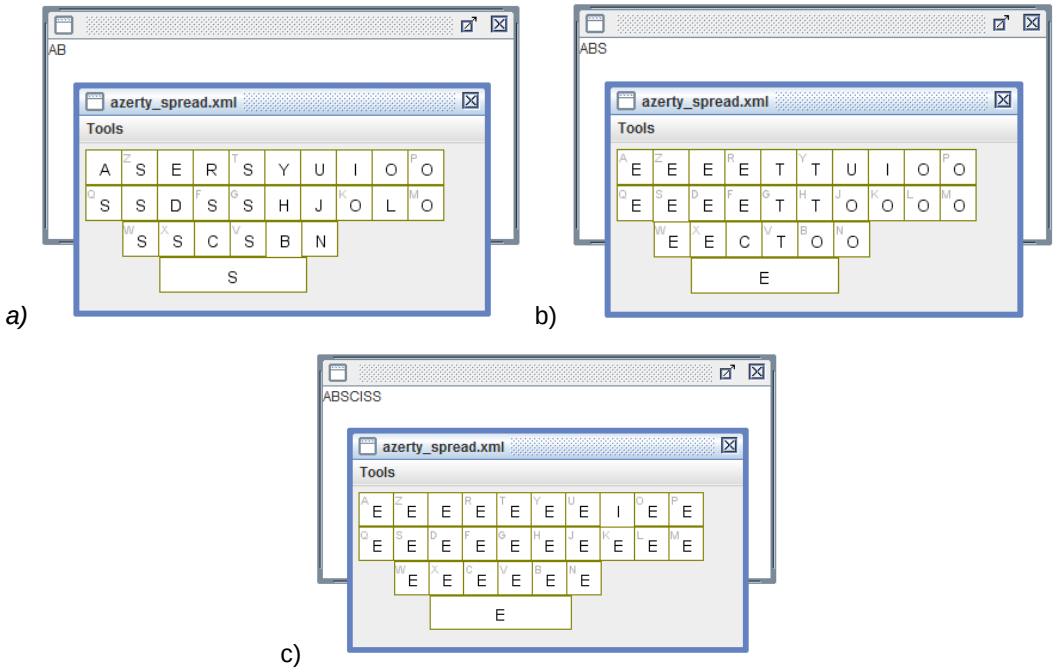


Figure 51. Exemple de recyclage des caractères

SpreadKey se destine a priori à deux types d'usage :

- un usage par des utilisateurs « lambda » sur un support mobile, via un écran tactile, dans un contexte d'interaction dégradée (mobilité par exemple) où la précision du pointage est rendue difficile par la petite taille des touches des supports mobiles et le déplacement ;

- un usage via un trackball ou une souris par des utilisateurs ayant un handicap moteur des membres supérieurs. Pour cette classe de population, la précision du pointage est contrariée par les propriétés motrices des usagers et la réduction des distances parcourues par le pointeur leur procure un confort.

En fonction, de ces deux contextes d'usage, l'impact du nombre de touches recyclées ou/et privilégiées et l'impact du coût de l'interaction alternative pour saisir le caractère originel d'une touche recyclée ne sont pas forcément identiques. En effet, un utilisateur plus lent sera proportionnellement moins pénalisé par l'introduction d'un coût fixe lié à l'interaction alternative. En conséquence, de manière à étudier les propriétés mécaniques de SpreadKey, et déterminer les paramètres P et R théoriquement idéaux pour ces deux contextes, nous avons procédé à des simulations sur la base des modèles théoriques.

Section II - Simulation et propriétés mécaniques de SpreadKey

Dans la mesure où SpreadKey évolue dynamiquement, il était impossible d'évaluer les performances à partir d'un modèle statique tel que le modèle strict de Soukoreff et MacKenzie [Soukoreff 95]. Nous avons donc procédé à des simulations de la saisie de textes via l'outil de simulation de la plate-forme E-Assist II.

Les simulations avaient pour objectif d'étudier l'impact des paramètres P et R par rapport à plusieurs variables : une estimation du temps de saisie (à partir de la loi de Fitts), une estimation de la distance minimum parcourue par le pointeur nécessaire pour procéder à la saisie, le nombre de touches recyclées et privilégiées, et enfin la taille moyenne des agrégations de touches pointées (proportionnelle au nombre de touches connexes[51] à la touche pointée et présentant le même caractère que la touche pointée).

II.1. Calcul théorique du temps de pointage

II.1.a) Procédure

Dans la mesure où plusieurs touches, ou agrégations de touches, présentent éventuellement le même caractère, la saisie d'une séquence de caractères peut être effectuée de manières complètement différente. Nous pouvions exploiter les deux stratégies proposées par l'outil de simulation de EAssist II : une stratégie consistant à minimiser à chaque fois le temps de pointage ; et une stratégie consistant à minimiser à chaque fois la distance parcourue par le pointeur. Néanmoins, dans la mesure où, en pratique, les résultats ne se sont pas montrés significativement différents, les résultats seront commentés uniquement en fonction de la première stratégie.

II.1.b) Variables mesurées

Pour l'estimation du temps de pointage, nous utiliserons la loi de Fitts :

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

- Avec W, (en posant C le nombre de touches connexes à la touche pointée présentant le même caractère que la touche pointée et W' la taille d'une touche) :

$$W = W'\sqrt{C}$$

- Ou D_{ij} est la distance entre le précédant pointage et le barycentre des touches connexes.

[51] Deux touches sont considérées connexes si elles partagent une partie de leur contour

- Pour les paramètres a et b, nous utiliserons les valeurs proposées par Soukoreff et MacKenzie [Soukoreff 95] (a=0 et b=1/4,9). Néanmoins, ces simulations étant effectuées en considérant différents degrés de handicap moteur des utilisateurs, nous simulerons ce handicap par une variation du paramètre b entre 1/4,9 – utilisateur normal – et 1 – utilisateur ayant un fort handicap moteur des membres supérieurs. Le paramètre b est ainsi défini en fonction de la *bande passante* [Shannon 48, Card 83, Soukoreff 95] de l'utilisateur : b = 1 / *bande passante* (soit une bande passante de 4,9 pour un utilisateur normal et de 1 pour les utilisateurs les plus handicapés).

Dans le cas de la répétition de touches (absence d'une tâche de pointage), nous utiliserons le temps TR mesuré par Soukoreff tel que TR = 0.153s.

Lorsque le caractère devant être saisi est un caractère recyclé, le temps de pointage est grevé de 300ms, temps nécessaire pour exécuter une pression longue avec SpreadKey.

La version de SpreadKey, proposée dans l'annexe B, illustre les modifications effectuées dans le clavier SpreadKey, décrit en KeySpec, de manière à permettre ces simulations.

II.2. Textes simulés

Pour un motif technique complexe, nous ne pouvions pas exploiter la fonctionnalité du système de simulation consistant à simuler la saisie des mots d'un dictionnaire et, pour chaque mot, pondérer le résultat par la fréquence du mot dans la langue[52].

Nous avons donc élaboré un corpus de textes sur la base de la version numérique des journaux « Le Monde ». Chacun de ces textes contenait 1 million de caractères. La ponctuation, l'accentuation ainsi que les nombres ont été retirés du corpus. Les différents corpus ont été évalués par l'outil de préparation de E-Assist II de manière à confirmer qu'ils respectaient la fréquence des caractères, des bigrammes et des mots.

Le corpus a été divisés en deux et nous avons vérifié que les résultats étaient identiques pour chacun des deux corpus. Ce corpus a donc été estimé représentatif de la langue.

II.3. Résultats

Les simulations ont été effectuées pour un grand nombre de valeur pour R, P[53] et en simulant différents degrés de handicaps moteurs (par des variations du paramètre b de la loi de Fitts).

La Figure 52 illustre l'impact de la variation du paramètre R sur le nombre de touches recyclées et, en contrepartie, Figure 53, la nécessité pour l'utilisateur de saisir des caractères recyclés. Naturellement, lorsque R augmente, le nombre de caractères considérés comme improbables augmente et le nombre moyen de touches recyclées augmente proportionnellement également. En conséquence, la probabilité que l'utilisateur ait

[52] Dans la mesure où le calcul de certaines variables (taille des touches pointées, nombre de touches recyclées, nombre touches privilégiées) est effectué au sein du clavier et de manière indépendante du système de simulation, ces valeurs sont calculées uniquement lorsque la simulation d'un clic est effectuée. Elles ne seraient donc pas pondérées par la fréquence des mots. Chaque mot serait saisi une seule fois et les variables affectées également une fois par mot, et ce de manière indépendante de la fréquence du mot.

[53] Le pas pour la variation des valeurs testées a été adapté de manière à illustré les variations significatives des courbes. Le pas a été réduit lorsque la courbe marquait des inflexions et augmenté lorsque la variation était plus linéaire.

recours à un caractère considéré comme improbable augmente aussi et avec elle le nombre de fois où il est nécessaire de saisir un caractères recyclés.

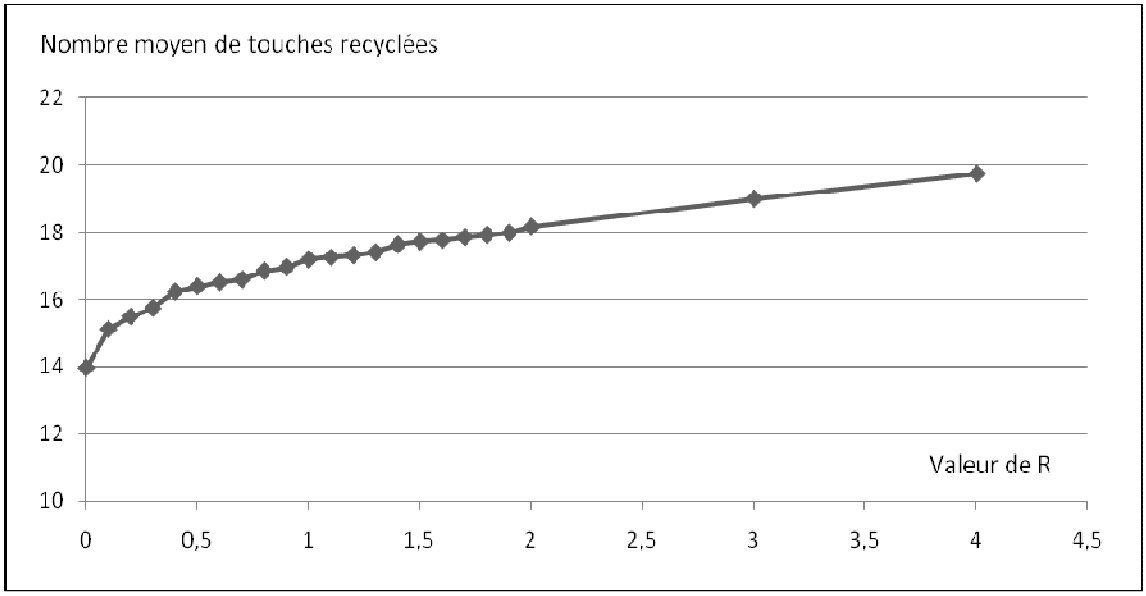


Figure 52. Nombre moyen de touches recyclées en fonction de R

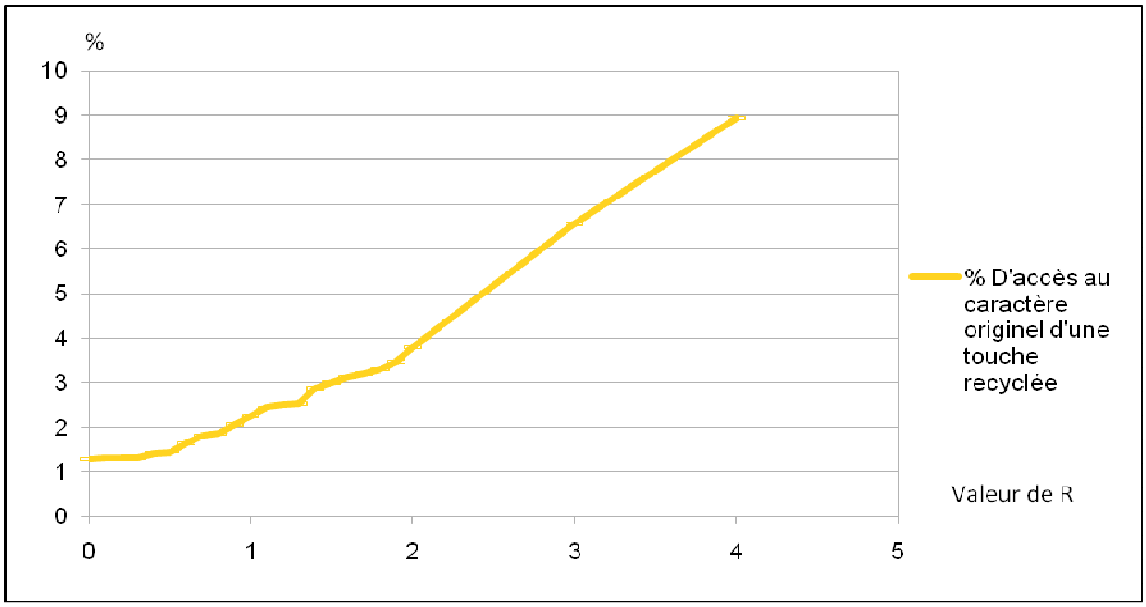


Figure 53. Pourcentage de caractères recyclés saisis en fonction de R

Ces deux facteurs corrélés – nombre de touches recyclées et nombre de caractères recyclés saisis – ont un impact antagoniste sur le temps de saisie. La hausse du nombre de caractères recyclés permet de rajouter des alternatives supplémentaires pour saisir les caractères les plus probables. En conséquence, ceci devrait permettre de réduire globalement les distances parcourues par le pointeur et d'accroître la taille moyenne des agrégations de touches pointées, et ainsi réduire le temps de pointage. La Figure 54 illustre ce résultat : nous considérons uniquement l'impact de l'augmentation du nombre de caractères recyclés en négligeant le coût d'accès supplémentaire à un caractère recyclé (ce coût est réduit à 0).

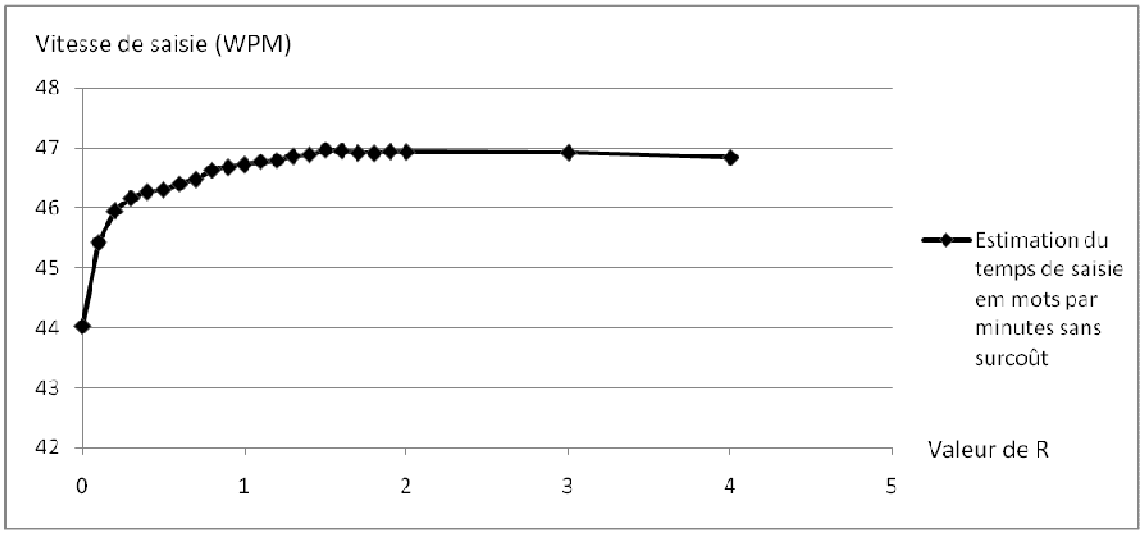


Figure 54. Evaluation théorique du temps de saisie en fonction de R en considérant 0ms comme surcoût pour l'accès à un caractère recyclé

En revanche, la hausse du nombre d'accès à des caractères recyclés, du fait que la saisie d'un caractère recyclé est plus longue que la saisie d'un caractère normal, devrait augmenter le temps moyen de saisie. La Figure 55 illustre une estimation du temps de saisie en rétablissant le surcoût dû à l'accès aux caractères recyclés. On peut observer que la perte de temps par rapport à la courbe de référence Figure 54 est proportionnelle au nombre de caractères recyclés saisis présentés en Figure 53.

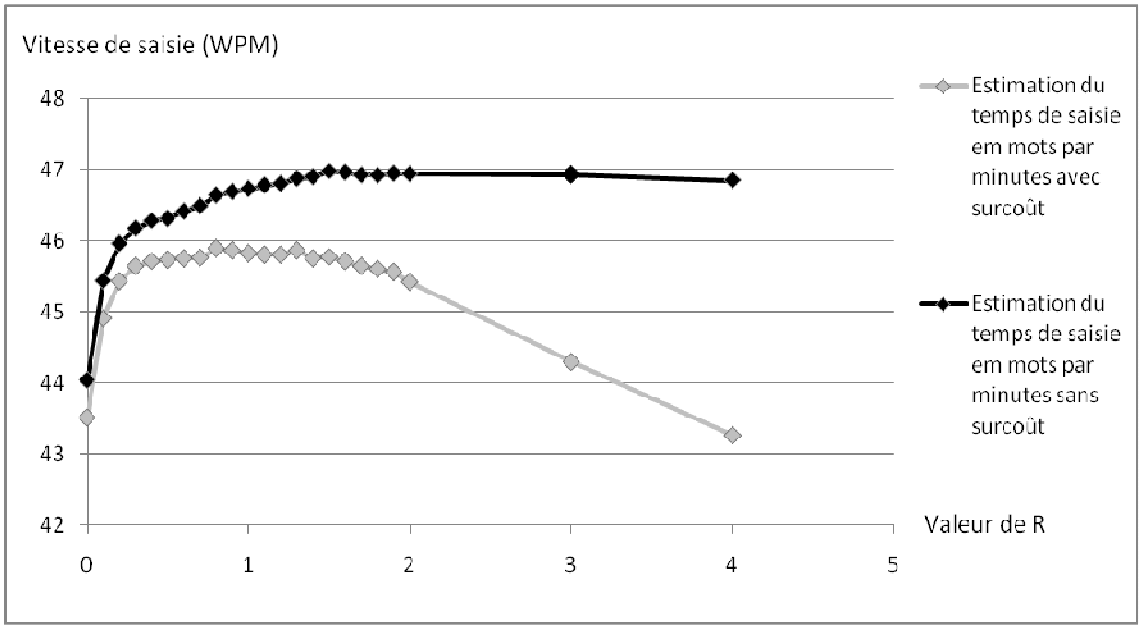


Figure 55. Evaluation théorique du temps de saisie en fonction de R en considérant 300ms comme surcoût pour l'accès à un caractère recyclé

A noter que les résultats présentés en Figure 55 sont calculés pour des utilisateurs normaux (b=1/4,9). Cependant, les deux facteurs n'auront pas forcément les mêmes influences sur un utilisateur normal que sur un utilisateur aux capacités motrices réduites. En effet, comme nous l'avons préalablement cité, le coût fixe introduit par la saisie d'un caractère recyclé sera

proportionnellement moins important pour un utilisateur handicapé moteur, plus lent à effectuer la tâche de pointage, que pour un utilisateur normal. Comme l'illustre la Figure 56, reprenant les mêmes simulations en considérant une bande passante de 3 pour l'utilisateur (donc avec une valeur de b=1/3), cette caractéristique aura tendance à déplacer le sommet de la courbe vers la droite lorsque la valeur de b augmente. Autrement dit, théoriquement, les utilisateurs handicapés devraient être plus favorisés par un plus grand nombre de caractères recyclés quitte à ce que cette hausse de P augmente les situations où il est nécessaire de saisir un caractère recyclé.

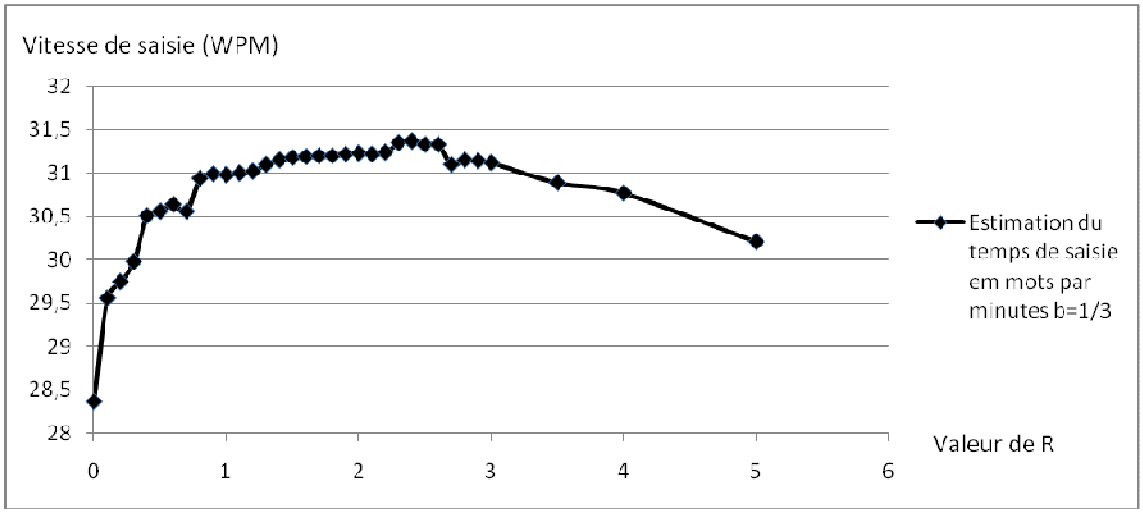


Figure 56. Evaluation théorique du temps de saisie en fonction de R en considérant 300ms comme surcoût pour l'accès à un caractère recyclé et une valeur de b=1/3

La valeur de P impacte le ratio *nombre de touches recyclées* par caractère privilégié ainsi que la probabilité pour un utilisateur de cibler l'un de ces caractères privilégiés. Une valeur basse pour P augmente le nombre de caractères privilégiés et en conséquence augmente la probabilité pour un utilisateur de cibler un caractère privilégié (cf. Figure 57). En contrepartie, les touches recyclées se répartissent entre les différents caractères privilégiés. En conséquence, plus il y a de caractères privilégiés, moins le nombre de touches recyclées par caractères privilégiés est élevé et plus les agrégations de touches sont de petite taille (Figure 58).

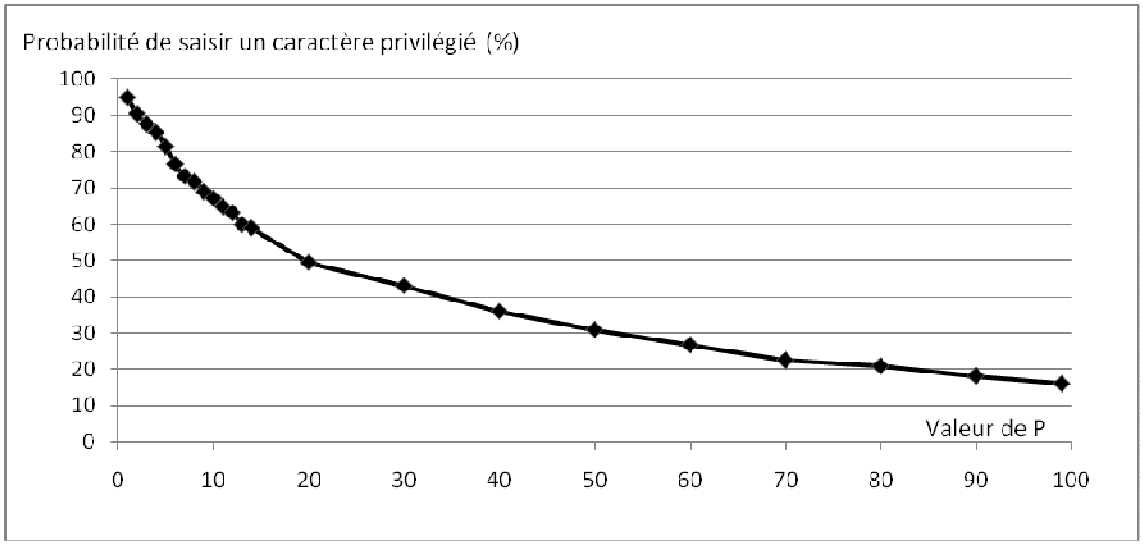


Figure 57. Probabilité de saisir un caractère privilégié en fonction de P pour R=0

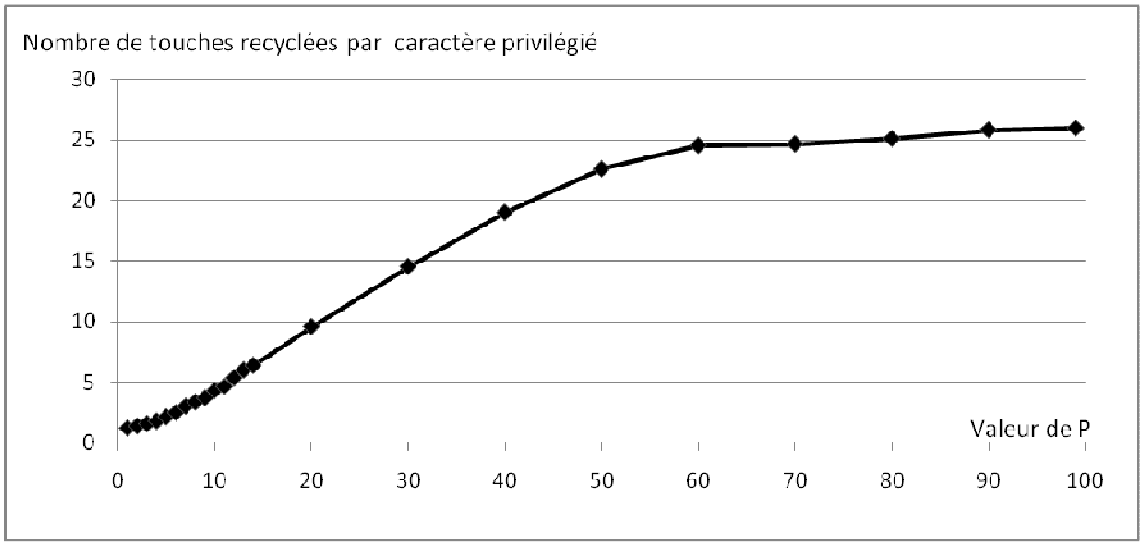


Figure 58. Nombre de touches recyclées par caractère privilégié en fonction de P et R=0,8

Là encore ces deux facteurs corrélés ont une influence contraire sur le temps de saisie. Une probabilité plus importante de saisir un caractère privilégié constitue une probabilité plus importante de saisir une touche de taille supérieure à une simple touche, avec pour conséquence de réduire le temps de saisie. En contrepartie, l'augmentation de la taille des touches privilégiées est moindre lorsqu'un plus grand nombre de caractères privilégiés se partagent les touches recyclées. Par conséquent, théoriquement d'après la loi de Fitts, si les touches sont moins grandes, la vitesse de saisie devrait augmenter de façon moins significative.

Néanmoins, les simulations montrent qu'un grand nombre de caractères privilégiés, soit une valeur faible de P, semble améliorer globalement les performances (cf. Figure 59) avec un maximum de 46.5 WPM pour P=5% (R fixé à 0.8%).

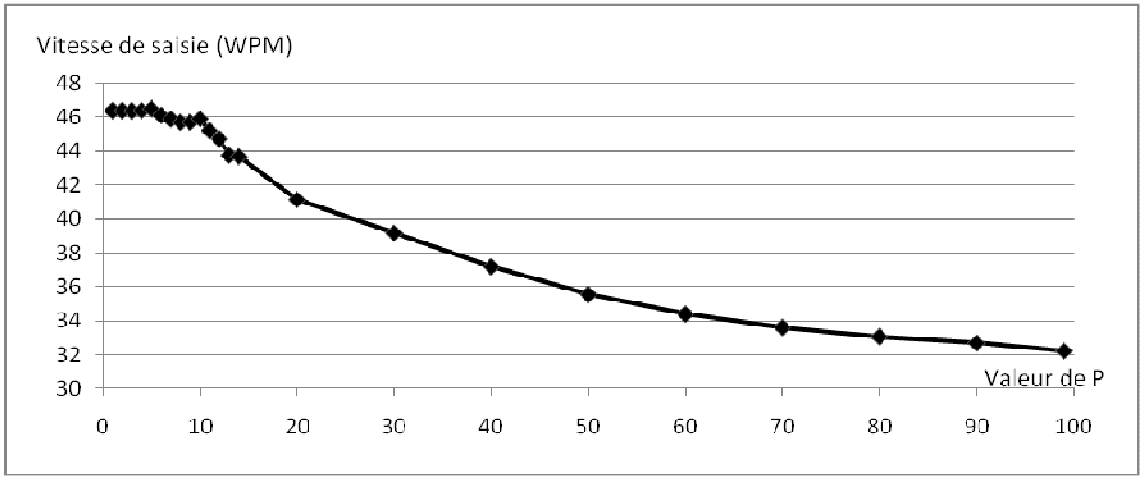


Figure 59. Evaluation théorique du temps de saisie en fonction de P et R=0,8

Paralèllement, le minimum (135pxs) en terme de distance parcourue par caractère par le pointeur est obtenu pour cette même valeur de P=5% (cf. Figure 60)[54]. La distance inférieure de 30% à la distance nécessaire pour le clavier AZERTY (194pxs).

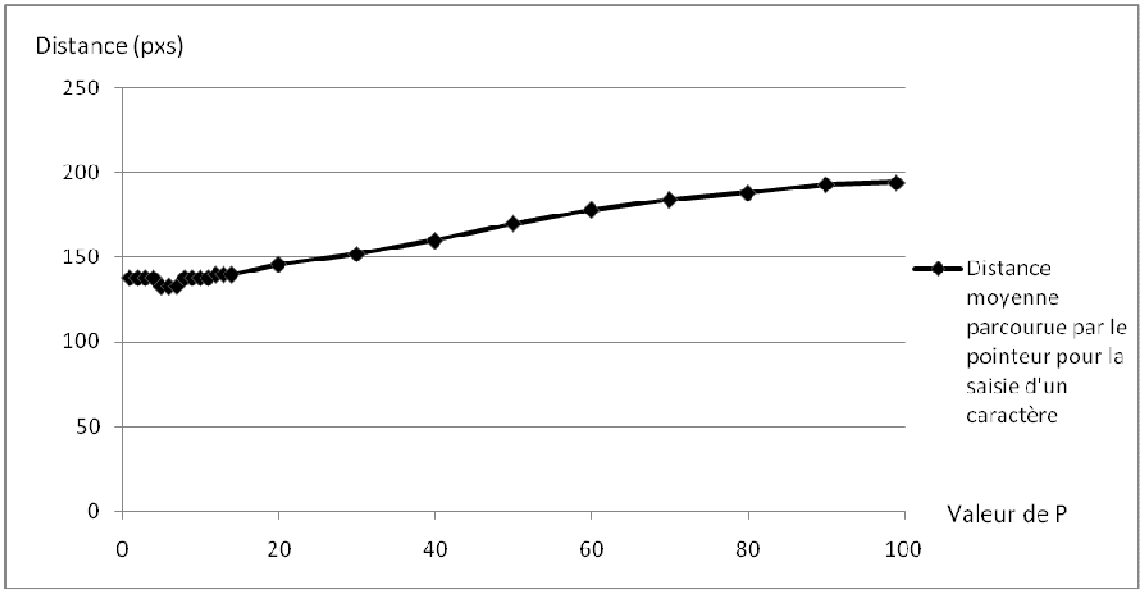


Figure 60. Distance moyenne parcourue par le pointeur pour la saisie d'un caractère en fonction de P et R=0,8

En conséquence, sur la base des évaluations théoriques, les performances optimales devraient être obtenues pour P=5% et une valeur de R dépendant du handicap de l'utilisateur. Ces performances sont résumées dans le tableau suivant (cf. Tableau 8). A noter que la vitesse de saisie est à titre indicatif et n'est pas dans l'absolu représentative des performances d'un utilisateur. La finalité de ces valeurs a pour unique objectif de comparer les différentes instances de SpreadKey sur un plan strictement mécanique. En effet, ici les coûts cognitifs sont ignorés et ne sont considérées que les temps relatifs à une utilisation mécanique du clavier.

Bande passante: (b = 1 / bande passante)	2	3	4,9
R (%)	2.4	1.9	0.8
Vitesse de saisie avec SpreadKey (wpm)	26.42	31.37	46.47
Vitesse de saisie avec AZERTY (wpm)	11.79	17.64	29.22

Tableau 8. Valeur de R optimale et vitesse de saisie théoriques en fonction des capacités motrices de l'utilisateur

[54] A noter que, en exploitant la stratégie de simulation consistant à minimiser systématiquement la distance, la distance minimale est également obtenue pour P=5% avec une valeur de 133pxs parcouru par caractère. La vitesse de saisie serait elle de 44.3wpm.

Section III - Évaluation heuristique de SpreadKey

L'évaluation heuristique de SpreadKey illustre que l'utilisateur ne devrait pas, a priori, être excessivement perturbé par les changements dynamiques, dans la mesure où il reste en mesure d'anticiper le déplacement en direction de la frappe suivante.

Néanmoins, les critères envisagés dans le cadre de cette évaluation heuristiques ne permettent pas d'identifier si l'utilisateur va conserver ou non la perception de la disposition AZERTY du clavier au fil des changements dynamiques. Ainsi, le clavier a implicitement été élaboré pour répondre de façon efficace (et peut-être artificielle) aux critères de la grille d'évaluation heuristique.

En tout état de cause, le clavier devrait s'avérer plus efficace pour des utilisateurs handicapés moteurs ou dans un contexte de mobilité, où la précision de l'utilisateur est altérée par le mouvement.

Section IV - Pré-expérimentation

Les évaluations théoriques avaient pour objectif d'étudier les propriétés mécaniques du clavier et de déterminer les meilleurs coefficients pour P et R. En revanche, en raison des nombreux changements dynamiques opérés sur le clavier au cours de la saisie, bien que ces changements soient effectués de manière à pertuber le moins possible l'utilisateur, il est difficile d'anticiper avec précision les coûts cognitifs qui vont venir grever l'usage et la vitesse de saisie. Les valeurs obtenues en termes de vitesse de saisie restent donc sujettes à caution et sont principalement destinées à comparer différentes instances de SpreadKey.

En conséquence, nous avons initié une pré-expérimentation destinée à explorer l'usage de SpreadKey par différents types d'utilisateurs handicapés moteurs ou non. Cette pré-expérimentation avait pour objectif d'obtenir une première estimation des performances relativement à différentes populations d'utilisateurs.

IV.1. Dispositif

La pré-expérimentation a été réalisée à travers la plate-forme E-Assist II. Les utilisateurs interagissaient à travers un ordinateur fixe au moyen de leur dispositif de pointage habituel (souris pour les utilisateurs les plus rapides et trackball pour les utilisateurs au handicap moteur plus marqué).

Les pré-expérimentations ont été effectuées avant l'exploitation des résultats de la simulation avec R=0% et P=12%. Les valeurs pour ces paramètres sont donc théoriquement sous-optimales.

IV.2. Participants

Nous avons recruté 10 volontaires pour cette pré-expérimentation. Parmi ces volontaires, 8 participants avaient un niveau plus ou moins marqué de handicap physique des membres supérieurs. La mesure de ce handicap sera discutée lors de la présentation des résultats.

IV.3. Tâche et stimulii

Deux jeux de données ont été créés manuellement. Chaque jeu de données contient 30 phrases courtes (environ 600 caractères). Les phrases respectent la fréquence des caractères et de bigramme de la langue avec un degré élevé de corrélation (97% pour la fréquence des caractères et 91% pour la fréquence des bigrammes).

La simulation de la saisie des deux jeux de données ne présente pas de différence significative en termes de temps de saisie et de distance parcourue par le pointeur (différence inférieure à 0,2% dans les deux cas) que ce soit pour la saisie via SpreadKey ou via AZERTY.

Par ailleurs, les deux jeux de données contenaient le même nombre de situations où un caractère recyclé devait être saisi.

La tâche consistait à recopier les phrases courtes des jeux de données au moyen du clavier logiciel relatif à l'exercice (AZERTY ou SpreadKey). Le texte à saisir était présenté dans un bandeau situé au-dessus du texte déjà saisi (cf. Figure 61). L'exercice était initié par une pression sur la touche espace. Une pression sur la touche espace permettait par ailleurs de passer à la phrase suivante autorisant l'utilisateur à effectuer une courte pause non comptabilisée entre deux phrases.

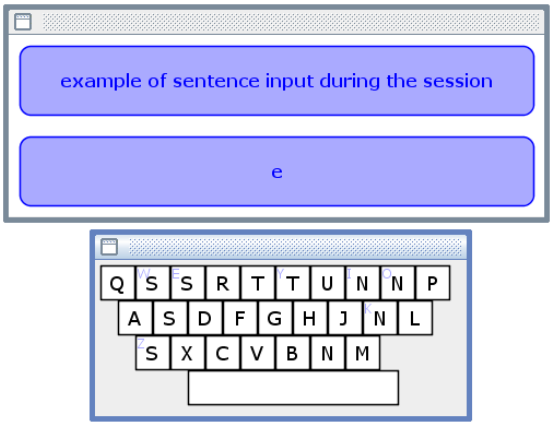


Figure 61. Configuration de la pré-expérimentation SpreadKey

Les erreurs de saisie n'étaient pas affichées à l'écran mais notifiées par un feedback sonore et visuel. Le texte saisi restait inchangé jusqu'à ce que l'utilisateur saisisse le caractère correct.

IV.4. Procédure

L'expérimentation était composée de 3 exercices : un exercice d'entraînement avec SpreadKey consistant à saisir 10 phrases courtes issues d'un autre jeu de données que ceux mentionnés ci-dessus, un second exercice avec SpreadKey et un exercice avec AZERTY. Avant chaque exercice un bandeau de consignes décrivait les instructions relatives à la nature de l'exercice et au clavier utilisé[55]. Une pause de trois minutes était appliquée entre l'exercice d'entraînement et le premier exercice, et de cinq minutes entre les deux exercices. Cette pause avait pour objet de neutraliser les effets de fatigue.

Dans la mesure où tous les utilisateurs sont déjà des usagers quotidiens du clavier AZERTY, l'ordre des exercices ne devrait pas avoir de conséquence en termes d'apprentissage. Néanmoins, les participants ont été divisés en deux groupes, l'un initiant par l'exercice avec AZERTY, l'autre avec SpreadKey. Le premier jeu de données était systématiquement utilisé pour le premier exercice (donc en alternance avec le clavier AZERTY et SpreadKey) et le second pour le second exercice.

[55] Cf. http://www.irit.fr/~Bruno.Merlin/research/index.php?expe=spreadey pour une version en ligne de l'expérimentation

L'expérimentation était accompagnée d'un petit questionnaire relevant : la perception de l'utilisateur sur ces performances (erreurs et temps de saisie) au moyen des deux claviers, leur perception sur les efforts physiques et cognitifs requis pour l'usage des différents claviers, leur perspective d'adoption de tel ou tel clavier pour un usage quotidien, et recueillant enfin d'éventuels commentaires libres.

Quatre sujets, les deux sujets au handicap physique le plus marqué et les deux sujets non handicapés, ont prolongé l'expérimentation sur dix sessions au rythme de 2 sessions par jour sur cinq jours. Entre chaque session, l'ordre des exercices était inversé.

IV.5. Résultats

L'étude des résultats n'a pas permis d'identifier d'éventuels effets d'apprentissages, de fatigue, ou un impact des jeux de données.

En raison de l'hétérogénéité des handicaps physiques entre les participants, les résultats sont présentés en fonction d'une estimation des capacités motrices des utilisateurs (de leur bande passante motrice). Cette estimation est basée sur les performances de l'utilisateur avec le clavier AZERTY. Nous considérons que l'utilisateur le plus rapide a une bande passante de 4,9 (modèle pour un utilisateur normal). La bande passante des autres utilisateurs est calculée par triangulation en fonction des performances avec AZERTY :

$$\boldsymbol{bande\_passante}(\boldsymbol{x}) = 4{,}9 \times \boldsymbol{WPM}(\boldsymbol{x}) / \boldsymbol{WPM}(\boldsymbol{plus\_rapide})$$

où WPM (x) est la vitesse de saisie d'un utilisateur avec AZERTY.

IV.5.a. Résultats relatifs à la première session engageant les 10 utilisateurs

Au cours de cette session, nous avons pu observer que les changements dynamiques opérés sur les claviers n'affectaient pas tous les utilisateurs de la même manière en fonction du degré de leur handicap (cf. Figure 62).

Les utilisateurs les plus rapides s'avéraient relativement désorientés par la succession des changements dynamiques et perdaient leur capacité à anticiper la saisie du caractère suivant. Ils étaient donc souvent conduits à analyser les changements opérés sur le clavier avant d'initier le pointage. Le gain obtenu en termes de pointage (relatif à l'augmentation de la taille des touches et à la réduction des distances parcourues par le pointeur) ne compensait pas le délai introduit par cette analyse. En conséquence, au cours de cette première session, les performances des utilisateurs les plus rapides se sont avérées inférieures avec SpreadKey qu'avec AZERTY.

Les utilisateurs les plus lents ont globalement adopté la même stratégie. En revanche, en raison d'un temps de pointage plus lent, ils ont été proportionnellement moins impactés par le temps d'analyse introduit. Le gain en termes de temps de pointage s'est donc avéré supérieur à la perte de temps relatif à l'analyse des changements dynamiques et les performances avec SpreadKey se sont ainsi révélées sensiblement supérieures aux performances avec AZERTY.

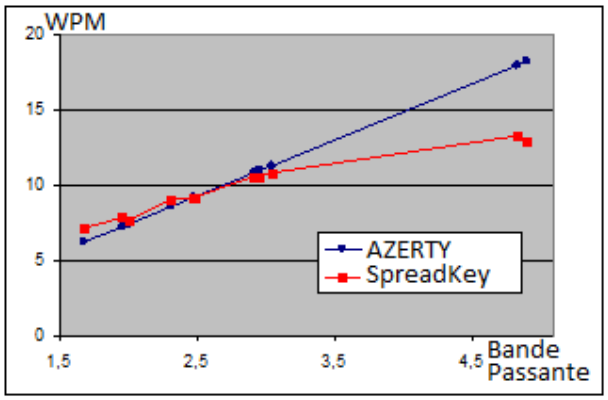


Figure 62. Vitesse de saisie de la première session en fonction d'une estimation de la bande passante des utilisateurs

Comme nous l'avions supposé au cours de l'évaluation théorique, SpreadKey s'avère donc plus efficient lorsque le handicap de l'utilisateur est plus élevé. Néanmoins, les résultats restent, au cours de cette première session, loin des valeurs théoriques calculées.

Comme nous pouvons l'observer Figure 63, le rapport entre la distance parcourue avec SpreadKey et la distance parcourue avec AZERTY est, en revanche, voisin des valeurs évaluées théoriquement.

Par ailleurs, nous avons pu observer que la distance parcourue par le pointeur avec AZERTY est supérieure pour les utilisateurs handicapés que pour les utilisateurs normaux (environ 7% supérieure à la distance prévue au cours des simulations pour les quatre utilisateurs les plus handicapés). Cette augmentation de la distance est notamment due à des oscillations du pointeur autour de la cible, révélatrices de difficultés à effectuer la tâche de pointage. En revanche, la distance parcourue avec SpreadKey est très voisine quelque soit la *bande passante* de l'utilisateur illustrant le bénéfice apporté par SpreadKey dans la tâche de pointage.

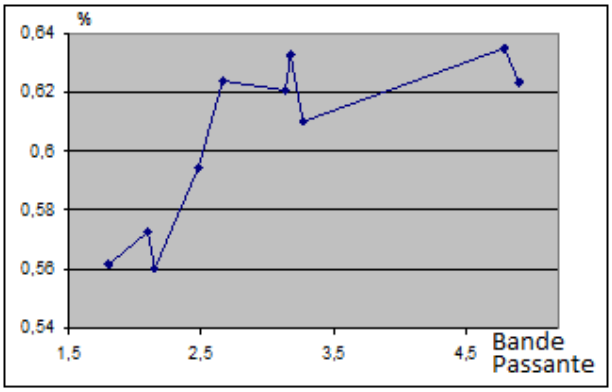


Figure 63. Ratio distance parcourue avec SpreadKey / distance parcourue avec AZERTY au cours de la première session en fonction de la bande passante des utilisateurs

L'analyse des erreurs montre une haute variabilité inter-individuelle : entre 1.2% et 4.2% d'erreurs avec AZERTY et entre 1.2% et 4.4% d'erreurs avec SpreadKey. Néanmoins nous pouvons observer un taux moyen d'erreurs supérieur de 1% avec SpreadKey pour les 6 utilisateurs les plus rapides.

IV.5.b. Étude de la progression des utilisateurs normaux

Les deux utilisateurs normaux ayant participé aux dix sessions étaient tous les deux propriétaires d'un téléphone à écran tactile et habitués à l'utilisation d'un clavier virtuel AZERTY. Leur progression (cf. Figure 64) au moyen du clavier AZERTY s'est en conséquence révélée faible et est plus à mettre sur le compte de l'apprentissage des phrases que du dispositif en lui-même.

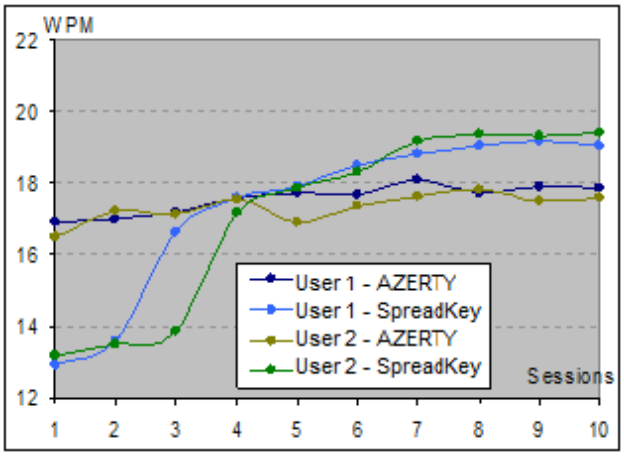


Figure 64. Progression des utilisateurs normaux

Sans surprise, l'apprentissage avec SpreadKey est nettement plus important. En revanche, cet apprentissage est marqué par une brutale inflexion entre les sessions 2 et 3 pour le premier utilisateur et entre les sessions 3 et 4 pour le second, avant de reprendre une forme logarithmique plus conventionnelle (cf. Figure 64). De façon parallèle à cette inflexion, la distance parcourue par le pointeur augmente d'environ 20% par rapport à la distance parcourue dans les sessions antérieures. Après consultation des utilisateurs, nous avons déduit que cette inflexion correspond à un changement de stratégie dans la manière d'utiliser le clavier.

Au cours des premières sessions, en raison des changements dynamiques opérés sur le clavier, les utilisateurs perdaient de vue l'agencement AZERTY des touches et avaient tendance à ne pas anticiper le mouvement du pointeur en direction de la prochaine touche (sauf si le caractère ciblé était dans le voisinage du précédent). Ils marquaient donc une pause pour analyser les changements dynamiques opérés et, dans un second temps, avaient plus tendance à pointer les touches à proximité du pointeur.

En revanche, au cours des sessions suivantes, le clavier a été utilisé d'une manière plus préconisée. L'utilisateur anticipait le déplacement en direction de la touche où devait se situer naturellement le caractère, tout en effectuant parallèlement l'analyse des changements dynamiques. La trajectoire du pointeur était opportunément infléchie en fonction de cette analyse. Par ailleurs, dans de nombreux cas, l'utilisateur anticipait le comportement de l'algorithme de recyclage (notamment sur les terminaisons de mots lorsque le clavier est recomposé en une ou deux grosses touches).

Néanmoins, même si à partir de la quatrième session la vitesse de saisie atteint puis dépasse la vitesse de saisie au moyen du clavier AZERTY, les deux utilisateurs ont soulignés un haut niveau de concentration nécessaire à l'utilisation du clavier. Le taux d'erreurs, supérieur de 1% au taux d'erreurs avec AZERTY, conforte cette appréciation qualitative. En outre, les deux utilisateurs ont insisté sur la difficulté à identifier les groupes

de touches comportant le même caractère et explicité que, dans certains contextes, l'anticipation du comportement de l'algorithme conduisait à des erreurs de saisie[56].

IV.5.c. Étude de la progression des utilisateurs handicapés moteur

La courbe de progression des utilisateurs handicapés moteurs (cf. Figure 65) se stabilise relativement rapidement (avec les deux claviers) sans marquer d'inflexion notable. Par ailleurs, la distance parcourue par le curseur reste stable au cours des sessions (avec une différence d'environ 0.8%).

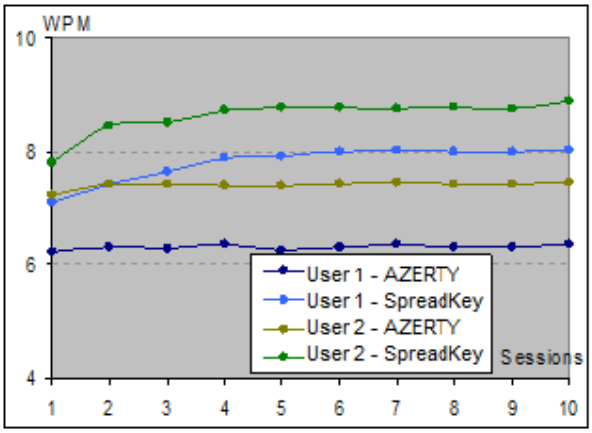


Figure 65. Progression des utilisateurs handicapés moteurs

Les utilisateurs ont confirmé que les pointages sont fastidieux et fatigants et, en conséquence, avoir privilégié, avec SpreadKey, la réduction du nombre de pointages et des distances de pointage à une éventuelle optimisation du temps de saisie. Les utilisateurs ont jugé le clavier globalement satisfaisant car, dans de nombreuses situations, la touche située immédiatement en dessous du curseur était recyclée avec le caractère souhaité. En conséquence, la saisie ne nécessitait pas de pointage. De même que les utilisateurs normaux, ils ont mentionné que, dans de nombreux cas et notamment pour la terminaison des mots longs, cette situation était prévisible, ce qui permettait de procéder à plusieurs clics successifs sans attendre l'analyse des changements (avec néanmoins quelques cas d'erreurs).

Les utilisateurs handicapés n'ont pas explicitement mentionné une fatigue cognitive relative à l'usage du clavier. Néanmoins, la difficulté à identifier les groupes de touches comportant le même caractère a également été soulignée.

IV.5.d. Conclusion de la pré-expérimentation

La pré-expérimentation a confirmé que SpreadKey était plus adapté pour les utilisateurs handicapés moteurs que pour des utilisateurs normaux, au moins dans le contexte d'usage étudié. Malgré des propriétés mécaniques intéressantes relatives à la taille des touches et aux distances, en l'état, les utilisateurs les plus rapides sont trop pénalisés par les perturbations engendrées par les changements dynamiques.

Cependant, malgré la configuration utilisée (P=12 et R=0) qui n'est pas optimale relativement aux propriétés mécaniques, SpreadKey a permis aux utilisateurs handicapés moteurs de

[56] Par exemple, lorsque plusieurs fois consécutives tout le clavier est recyclé avec la même lettre et que subitement apparaissent plusieurs alternatives.

réduire la distance parcourue par le pointeur d'environ 45% tout en améliorant le temps de saisie d'environ 22%.

Section V - Expérimentation

L'objectif des expérimentations était d'approfondir l'étude de SpreadKey auprès d'une population homogène de sujets handicapés moteur des membres supérieurs. Conformément aux résultats des simulations et des pré-expérimentations, notre hypothèse est que SpreadKey devrait permettre de réduire les distances couvertes par le pointeur tout en améliorant la vitesse de saisie.

V.1. Participants

Six participants brésiliens (6 hommes), tous faisant état d'un sévère handicap moteur des membres supérieurs[57], ont participé à l'expérimentation. Les utilisateurs étaient tous des usagers réguliers d'un clavier logiciel QWERTY et interagissaient avec le clavier via un trackball.

Aucun des participants n'avait préalablement participé à la pré-expérimentation.

V.2. Dispositif

Le dispositif d'expérimentation est identique au dispositif exploité au cours de la pré-expérimentation. Néanmoins, les paramètres P et R de SpreadKey ont été corrigés en fonction des résultats de l'analyse théorique. Les valeurs configurées sont P=5% et R=2.4% théoriquement optimales pour l'usage par des utilisateurs handicapés moteurs.

V.3. Tâche et stimulii

Un jeu de données a été élaboré pour le portugais en respectant la méthodologie mise en œuvre pour la création des jeux de données de la pré-expérimentation.

Les utilisateurs avaient pour tâche de recopier les phrases contenues dans le jeu de données. Les phrases étaient identiques pour les deux exercices et pour chaque session. Les utilisateurs étaient engagés à réaliser la tâche le plus efficacement possible. Les phrases à recopier étaient présentées dans un bandeau situé au-dessus d'un bandeau contenant les caractères déjà saisis (cf. Figure 61).

Les erreurs n'étaient pas affichées ni corrigées, mais signalées par un feedback sonore et visuel. Le bandeau contenant les caractères saisis restait inchangé jusqu'à ce que l'utilisateur presse le caractère correct.

V.4. Procédure

Chaque sujet a réalisé 10 sessions sur une période de 5 jours : deux sessions par jour espacées de 3h à 5h. Chaque session consistait en deux exercices de recopie : l'un au moyen d'un clavier QWERTY restreint aux 26 caractères de l'alphabet plus l'espace, l'autre au moyen du clavier augmenté du système SpreadKey. Une courte session d'entraînement était réalisée pour chaque dispositif avant la première session.

Les utilisateurs étaient divisés en deux groupes initiant la première session par un artefact différent (QWERTY ou SpreadKey). Les utilisateurs alternaient l'ordre des exercices après

[57] Avec une bande passante évaluée entre 1.8 et 2.1 sur la base du calcul réalisé en Section III -

chaque session. Les phrases étaient identiques pour les deux exercices et pour chaque session.

Pour initier l'expérimentation et pour passer à la phrase suivante, l'utilisateur devait presser la touche espace.

Comme pour la pré-expérimentation, les variables indépendantes étudiées étaient les artefacts QWERTY ou SpreadKey et les variables dépendantes étaient la vitesse de saisie, la distance parcourue par le dispositif de pointage et le taux d'erreurs.

V.5. Résultats

Les données statistiques présentées dans cette section ont été étudiées à travers une ANOVA (analyse de la variance, covariance). L'ANOVA a en premier lieu montré que l'ordre des exercices n'avait pas d'incidence sur les résultats et sur leur validité.

V.5.a. Étude de la distance parcourue par le pointeur

En moyenne, la distance parcourue par le pointeur était de 213pxs pour le clavier AZERTY contre 138pxs pour SpreadKey (cf. Figure 66). En conséquence, SpreadKey a permis de réduire la distance d'environ 35% ($F(1, 2398)=3227.3$ et $P<0.0001$). La distance est 8.5% supérieure à la distance envisagée au cours des simulations pour le clavier AZERTY et 3.6% supérieure pour SpreadKey. En conséquence, le bénéfice est lui supérieur de 35% avec SpreadKey contre 30% envisagé en simulation. L'expérimentation a ainsi confirmé que, en réduisant les distances et en augmentant la taille des touches, SpreadKey permettait de faciliter le pointage et de réduire les oscillations autour de la cible.

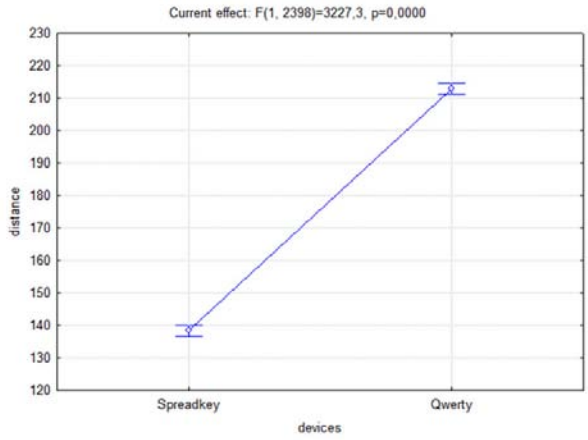


Figure 66. Moyenne des distances parcourues

V.5.b. Étude de la vitesse de saisie

En moyenne, les participants ont saisi au rythme de 0,87 caractères par seconde (soit 10,44wpm) au moyen du clavier QWERTY, et au rythme de 1,09 caractères par seconde avec SpreadKey (soit 13,08wpm). L'expérimentation a donc confirmé un gain de temps d'environ 20% avec SpreadKey (cf. Figure 67).

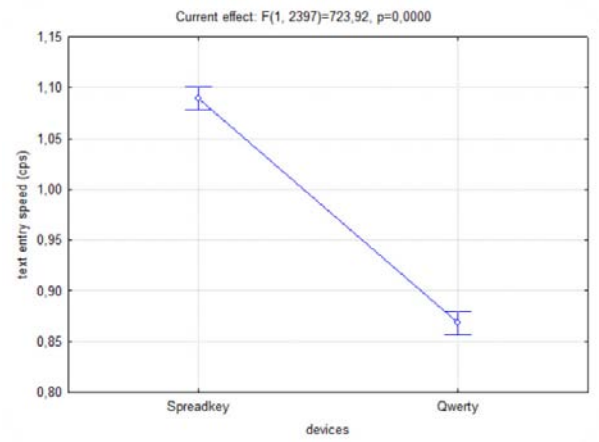
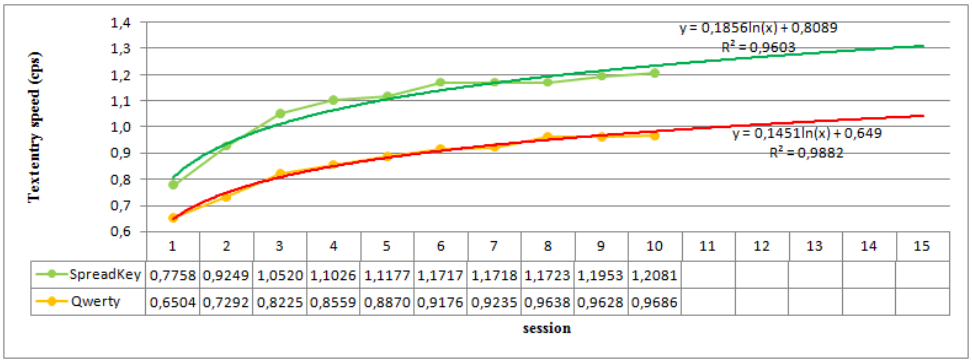


Figure 67. Moyenne des temps de saisie

A noter que, dès la première session, les utilisateurs se sont avérés plus efficaces avec SpreadKey qu'avec QWERTY. L'écart a eu tendance à s'accentuer en faveur de SpreadKey avec le temps (cf. Figure 68).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SpreadKey	0,7758	0,9249	1,0520	1,1026	1,1177	1,1717	1,1718	1,1723	1,1953	1,2081					
Qwerty	0,6504	0,7292	0,8225	0,8559	0,8870	0,9176	0,9235	0,9638	0,9628	0,9686					

Figure 68: Progression des utilisateurs

Les observations des utilisateurs faites au cours des pré-expérimentations, relatives à la stratégie d'utilisation et au design de SpreadKey, ont également été confirmées. Les utilisateurs ont confirmé préférer ne pas anticiper le déplacement en direction de la prochaine touche de manière à réduire le nombre et la distance des pointages au détriment d'une meilleure efficacité de la saisie. Ils ont également souligné la difficulté d'identifier les groupes de touches contenant le même caractère.

V.5.c. Étude du taux d'erreur

Dans la mesure où SpreadKey augmente la taille des touches et réduit les distances parcourues par le pointeur, nous pouvions espérer un taux d'erreur inférieur avec SpreadKey qu'avec QWERTY. Comme pour la pré-expérimentation, les résultats ont infirmé cette hypothèse. La moyenne du taux d'erreur fut de 1.5% avec QWERTY contre 2.6% avec SpreadKey. La différence s'est par ailleurs accentuée de 0.2% entre la pré-expérimentation et l'expérimentation.

Les commentaires des utilisateurs, confirmés par une analyse plus approfondie des erreurs ont montré qu'un grand nombre d'erreurs étaient dues à l'anticipation de l'utilisateur par rapport au résultat de l'algorithme de recyclage. Une seconde source d'erreurs est relative à des erreurs d'interaction pour saisir un caractère recyclé. Le nombre de caractères recyclés était supérieur à celui de la pré-expérimentation en raison de la valeur du paramètre R choisi

(R=2,4 contre 0 au cours de la pré-expérimentation). En conséquence, les utilisateurs devaient faire plus souvent appel à l'interaction alternative pour sélectionner le caractère.

Les utilisateurs ont par ailleurs exprimé explicitement une gêne par rapport à la nécessité de saisir de nombreux caractères recyclés. Cette gêne s'exprimait principalement en termes de recherche des caractères : au cours de la saisie d'un mot peu fréquent mais néanmoins commun, un caractère recyclé était spontanément visuellement recherché parmi les caractères non recyclés ou privilégiés. Ces commentaires incitent donc à repenser l'idée d'une valeur de R élevée qui, si elle permet de meilleures performances théoriques du point de vue mécanique, engendre des perturbations du point de vue cognitif.

V.6. Conclusion

Les expérimentations ont donc confirmé statistiquement les tendances observées au cours des pré-expérimentations. Bien que les performances soient nettement inférieures aux valeurs calculées théoriquement en termes de vitesse de saisie, SpreadKey s'est néanmoins avéré 20% plus efficace que le clavier AZERTY et également plus compétitif qu'espéré en termes de réduction des distances.

Les observations des utilisateurs ont cependant montré que des efforts pouvaient être faits pour facilité la lisibilité des changements dynamiques opérés sur le clavier et diminuer les erreurs.

Section VI - Perspectives : SpreadKey II

Ces efforts ont fait l'objet d'une seconde version de SpreadKey, qui sera évaluée dans un futur proche. Cette deuxième version de SpreadKey[58] a pour objectif d'améliorer les performances des utilisateurs handicapés (essentiellement réduire les erreurs et augmenter le confort d'utilisation) et aborder un public d'utilisateurs valides sur les dispositifs mobiles.

SpreadKey II reprend globalement les concepts de la première version : recyclage dynamique des touches dont le caractère est improbable par un caractère à forte probabilité d'être saisi ; et interaction secondaire pour saisir un caractère recyclé. Néanmoins plusieurs aménagements ont été effectués de manière à simplifier la lecture des changements dynamiques opérés, réduire les erreurs, et simplifier leur correction :

- Réduction de la complexité des changements : les touches sont recyclées uniquement si elles sont connexes à la touche contenant le caractère qu'elles remplacent, ou connexes à une touche connexe. En conséquence, le nouveau clavier ne contient plus qu'une touche ou une agrégation de touches contenant le même caractère. Par exemple (cf. Figure 69), après la saisie du caractère « F », SpreadKey I contient deux agrégations de touches séparées contenant le caractère « O », en revanche SpreadKey II n'en contient plus qu'une ;

- Simplification de l'interaction : le paramètre R est réduit à 0 de manière à réduire au minimum les cas où l'interaction secondaire est nécessaire ;

- Faciliter l'identification des groupes de touches : les agrégations de touches contenant le même caractère sont rassemblées pour former une touche unique dont le fond est teinté d'une couleur distincte pour chaque groupe de touches (cf. Figure 69) ;

[58] Description KeySpec de « SpreadKey II » disponible en Annexe B

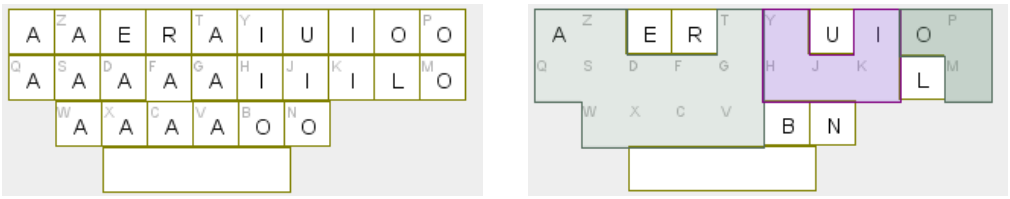


Figure 69. SpreadKey I (gauche) et SpreadKey II (droite) après la saisie du caractère « F »

- De manière à accélérer l'identification des groupes de touches contenant le caractère espace, ce groupe de touches est systématiquement teinté de la même couleur nettement distincte des autres couleurs utilisées (jaune, cf. Figure 70) ;

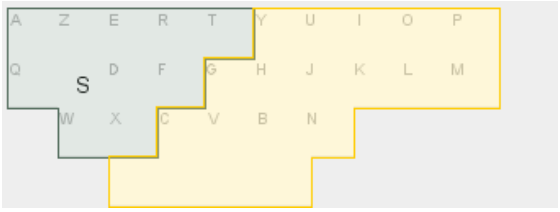


Figure 70. Les agrégations de touches contenant le caractère espace sont toujours de la même couleur, distincte des autres couleurs (jaune)

- Faciliter la correction des erreurs : une interaction supplémentaire, un simple trait en direction de la droite ou du bas, permet d'effacer le dernier caractère saisi.

Chapitre 11 - Claviers à pointage sémantique

Comme nous avons pu l'observer à travers les expérimentations de SpreadKey, la modification dynamique de l'espace visuel, même si elle permet parfois d'améliorer les performances, reste éprouvante cognitivement et est difficilement acceptée par l'utilisateur dans un contexte d'utilisation normal. En conséquence, la deuxième stratégie que nous avons étudiée avait pour objectif de modifier l'espace physique sans modifier l'espace visuel en conservant ainsi la scrupuleusement disposition originelle des caractères.

Nous avons exploré deux techniques distinctes : d'une part pour les claviers logiciels en environnement bureautique à destination des utilisateurs handicapés moteur ; et d'autre part sur plateformes mobiles à destination d'utilisateurs quelconques.

Dans un environnement bureautique, le pointeur est manipulé de façon indirecte via une souris ou un trackball. L'idée était, dans ce contexte, d'exploiter le pointage sémantique [Blanch 04], consistant à modifier la dynamique du déplacement du pointeur pour faciliter l'accès aux touches les plus probables (cf. I. SemanticKeyboard).

Sur les supports mobiles, le pointeur est manipulé de façon directe. Il n'est donc pas possible d'influer sur la dynamique de sa trajectoire. A l'image de [Aulagner 10] ou [Al Faraj 09a], il est donc possible de s'inspirer des techniques de zoom sémantique [Furnas 86] pour faciliter l'accès aux touches (cf. II. SemanticKeyboard Mobile).

Section I - SemanticKeyboard

Cette première version s'adresse donc à des utilisateurs handicapés moteurs. Elle s'inspire du concept de pointage sémantique [Blanch 04] fréquemment exploité pour faciliter l'atteinte de cibles graphiques par des utilisateurs tels que les enfants en bas âge [Hourcade 08], les personnes âgées [Hourcade 10] ou les utilisateurs handicapés moteurs [Findlater 10] pour qui les capacités motrices sont diminuées.

I.1. Concept

L'idée du pointage sémantique est d'accélérer le mouvement du pointeur lorsque celui-ci traverse des espaces vierges de cibles, d'objets graphiques interactifs et de ralentir le mouvement lorsque le pointeur s'approche et traverse des cibles, notamment petites. D'un point de vue moteur, le pointage sémantique permet donc : d'effectuer un effort moindre pour atteindre des cibles distantes et d'accroître virtuellement la taille des cibles dans la mesure où l'effort nécessaire pour traverser la cible est démultiplié.

En conséquence, l'idée de SemanticKeyboard est d'adapter l'espace moteur en fonction de la probabilité d'occurrence des caractères. L'espace moteur est modifié dynamiquement en fonction des caractères préalablement saisis. Plus un caractère est probable dans le contexte de la saisie, plus la taille de la touche est virtuellement augmentée, et donc plus l'effort nécessaire pour le traverser est important. A l'inverse, les caractères improbables sont rendus plus faciles à traverser. En conséquence, SemanticKeyboard rapproche les caractères à forte probabilité d'être saisis tout en augmentant la taille des touches ciblées, et ceci sans modifier l'aspect visuel du clavier.

I.2. Evaluation

Une première version de SemanticKeyboard a été évaluée expérimentalement. En raison des difficultés pour accéder à des utilisateurs handicapés, ces expérimentations ont été réalisées auprès d'utilisateurs valides. 12 utilisateurs volontaires, 10 hommes et 2 femmes étudiants ou chercheurs, tous droitiers et âgés de 21 à 44 ans, ont donc participé à cette expérimentation.

L'expérimentation consistait à effectuer deux exercices de recopie de 21 phrases (la première phrase servait d'entraînement et n'était pas décomptée). Un exercice était effectué au moyen du clavier AZERTY et l'autre au moyen du clavier AZERTY augmenté du pointage sémantique (SemanticKeyboard). Le pointeur était manipulé via une souris. Les phrases étaient tirées au sort parmi un jeu de 30 phrases courtes.

Les utilisateurs étaient répartis en deux groupes alternant l'ordre des exercices de manière à contrebalancer les effets de fatigue et d'apprentissage.

L'expérimentation était complétée par un questionnaire permettant aux participant d'évaluer les claviers de 1 (mauvais) à 7 (très efficient) du point de vue des 5 critères suivants : vitesse ; précision ; effort ; fatigue ; et confort.

I.3. Résultats

Les résultats numériques, obtenus au cours de cette session unique, montrent une certaine difficulté des utilisateurs à s'adapter aux changements dynamiques opérés sur la trajectoire du pointeur. Les gains de temps obtenus grâce à une réduction « virtuelle » des distances et à une augmentation « virtuelle » de la taille des touches sont contrebalancés par une difficulté initiale à suivre la trajectoire du pointeur, engendrant des délais et quelques erreurs supplémentaires. Au final, les temps de saisie avec les deux claviers se sont avérés sensiblement équivalent (10,2 wpm pour SemanticKeyboard contre 10,3 pour AZERTY) avec un taux d'erreurs légèrement supérieur pour SemanticKeyboard ($F(1,11) = 12.210$ et $p=0.005$ pour l'ANOVA).

On note néanmoins une progression significative au cours de la session laissant supposer de meilleures performances au terme d'un temps d'usage plus long (cf. Figure 71). Cette tendance à l'amélioration justifie une bonne appréciation qualitative du clavier relativement au critère de vitesse de saisie (cf. Figure 72).

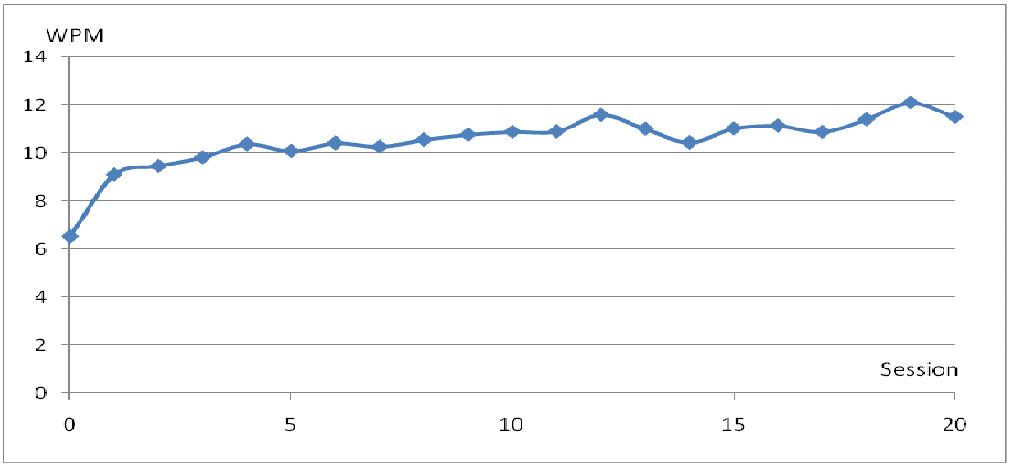


Figure 71. Progression au cours de la session (temps de saisie par phrase) avec SemanticKeyboard

Concernant les évaluations qualitatives, outre un sentiment supérieur d'efficacité en manière de rapidité, celles-ci soulignent un sentiment de confort et de diminution de la fatigue. En revanche, le système induit quelques erreurs qui portent préjudice au critère de précision.

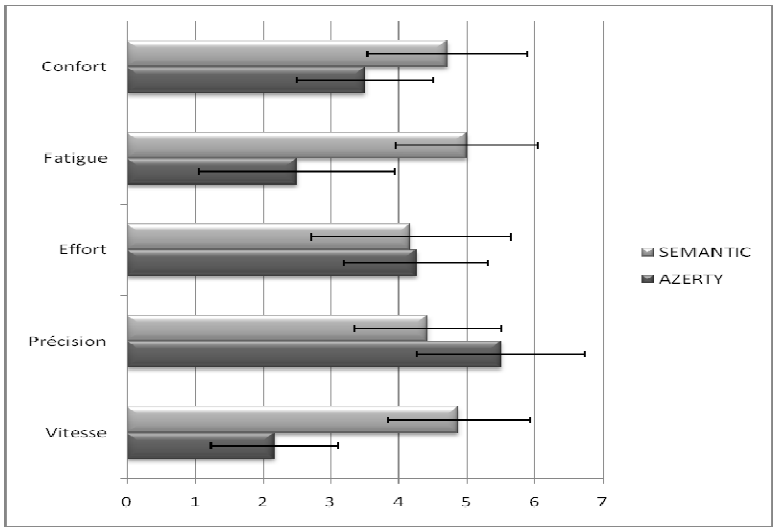


Figure 72. Évaluation qualitative

En conséquence, malgré des résultats numériques grevés par un temps d'adaptation nécessaire en début de session, les évaluations restent encourageantes. Néanmoins, un travail reste à fournir de manière à améliorer l'algorithme et rendre la trajectoire du curseur plus prédictible et plus facile à suivre.

Section II - SemanticKeyboard Mobile

Sur les supports mobiles, l'utilisateur manipule le pointeur (doigt ou stylet) de manière directe. Nous ne pouvons donc pas influer de façon logicielle sur cette trajectoire. La seule marge de manœuvre en termes de modification de l'espace moteur est d'influer sur la taille des touches.

Plusieurs projets tels que BigKey [Al Faraj 09a] et FloodKey [Aulagner 10] augmentent dynamiquement la taille des touches en fonction de la probabilité de saisie des caractères. BigKey est établi sur la base de touches initialement hexagonales, tandis que FloodKey est développé sur la base du layout de CATKey [GO 07]. Dans les deux cas, l'aspect visuel des claviers se transforme continuellement en fonction des modifications des probabilités d'occurrence des caractères.

Le principe de SemanticKeyboard dans sa version mobile reste de modifier l'espace physique sans modifier l'espace visuel. En analysant les distributions de frappes étudiées par Zhai [Zhai 02b] (cf. Figure 73), nous observons que l'utilisateur pointe assez systématiquement la partie centrale de la touche. En revanche, les petits écarts de mouvements conduisent assez facilement à atteindre la bordure de la touche voisine et à générer une erreur.

Nous observons par ailleurs que les écrans tactiles ont une certaine marge d'imprécision variant en fonction des modèles et de leur usure.

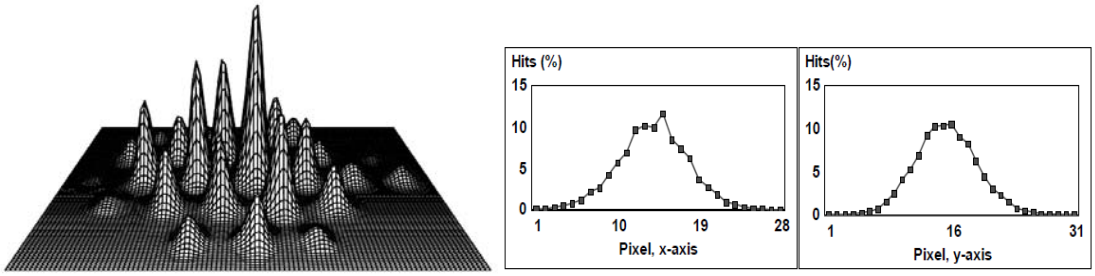


Figure 73. Distribution des frappes sur le clavier Metropolis [Zhai 02b]

Sur la base de ces deux observations, l'idée de SemanticKeyboard Mobile est que le doute – une frappe effectuée dans un espace à la frontière entre deux touches – devrait bénéficier au caractère le plus probable. Ainsi, plus la probabilité de frappe entre les deux touches est à l'avantage d'une touche, plus la marge est à l'avantage de cette touche. Le clavier augmente donc la taille des touches les plus probables mais sans que les altérations ne soient visibles à l'écran (cf. Figure 74).

La conséquence recherchée est la diminution des erreurs notamment en contexte de mobilité.

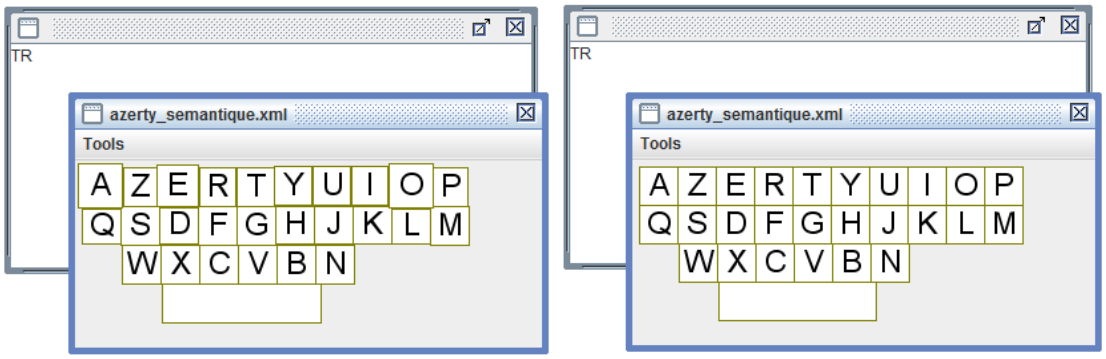


Figure 74. A gauche l'espace physique, à droite l'espace visuel

Les dimensions (et la position) de chaque touche sont modifiées asymétriquement en fonction de la probabilité des touches voisines.

Mathématiquement, les transformations opérées sur chaque touche s'expriment de la façon suivante :

- Nous considérons P, Ph, Pb, Pd et Pg les probabilités fournies par le système de prédiction de : la touche considérée (P), de la touche située en haut de cette touche (Ph), en bas (Pb), à droite (Pd) et à gauche (Pg).

- Dans le cas où une touche parmi ces quatre dernières n'existe pas, sa probabilité est réduite à 0 (Par exemple : si la touche considérée est A, Ph = Pg = 0).

- Nous nommerons par ailleurs T, Xo, Yo les variables représentant respectivement la taille initiale des touches (carrées, donc la hauteur est originellement égale à la hauteur) et les coordonnées d'origine de la touche considérée. D'autre part, X et Y seront les nouvelles coordonnées de la touche après transformation, et W et H les nouvelles largeur et hauteur de la touche.

Nous posons D_h, D_b, D_d, D_g tel que :

$$\boldsymbol{D_h} = \frac{\boldsymbol{T} \times (\boldsymbol{P} - \boldsymbol{P_h})}{2 \times 100}$$

$$\boldsymbol{D_b} = \frac{\boldsymbol{T} \times (\boldsymbol{P} - \boldsymbol{P_b})}{2 \times 100}$$

$$\boldsymbol{D_d} = \frac{\boldsymbol{T} \times (\boldsymbol{P} - \boldsymbol{P_d})}{2 \times 100}$$

$$\boldsymbol{D_g} = \frac{\boldsymbol{T} \times (\boldsymbol{P} - \boldsymbol{P_g})}{2 \times 100}$$

En conséquence, les dimensions et la position du clavier seront modifiées tel que :

$$\boldsymbol{X} = \boldsymbol{X_O} - \boldsymbol{D_g}$$

$$\boldsymbol{Y} = \boldsymbol{Y_O} - \boldsymbol{D_h}$$

$$\boldsymbol{W} = \boldsymbol{T} + \boldsymbol{D_d} + \boldsymbol{D_g}$$

$$\boldsymbol{H} = \boldsymbol{T} + \boldsymbol{D_h} + \boldsymbol{D_b}$$

Section III - Simulation

Les évaluations ont été conduites sur la base des jeux de données recueillis aux cours des expérimentations du clavier multi-layer (cf. Chapitre suivant). Ces jeux de données contiennent les interactions (mouvements et clics de la souris) avec le clavier multi-layer, pour les trois utilisateurs ayant terminé les expérimentations en effectuant entre 78 et 80 sessions d'environ 6 minutes chacune (soit un total d'environ 8h de saisie par utilisateur).

Bien que ces expérimentations ne se soient pas déroulées dans un contexte de mobilité, la tâche, compliquée par une très petite taille des touches, avait généré un grand nombre d'erreurs relatives à des imprécisions de la part des utilisateurs.

Les interactions effectuées par les utilisateurs, avec le clavier multi-layer, durant les différentes sessions de l'expérimentation[59], ont été rejouées sur le même clavier muni du système SemanticKeyboard Mobile[60].

[59] Les interactions sont mémorisées par le système de trace de la plateforme E-Assist II

[60] La distribution de touches du clavier SemanticKeyboard a été adaptée au cours des simulations des différentes sessions de manière à suivre les évolutions du clavier multilayer.

Les phrases résultant de la simulation ont ensuite été comparées aux phrases de référence recopiées au cours de l'expérimentation dans le but de mesurer l'impact du système sur le nombre des erreurs.

Les résultats ci-dessous présentent l'évolution des erreurs au cours des sessions de l'expérimentation du clavier multi-layer et, en parallèle, les erreurs mesurées au cours des simulations avec le système SemanticKeyboard Mobile.

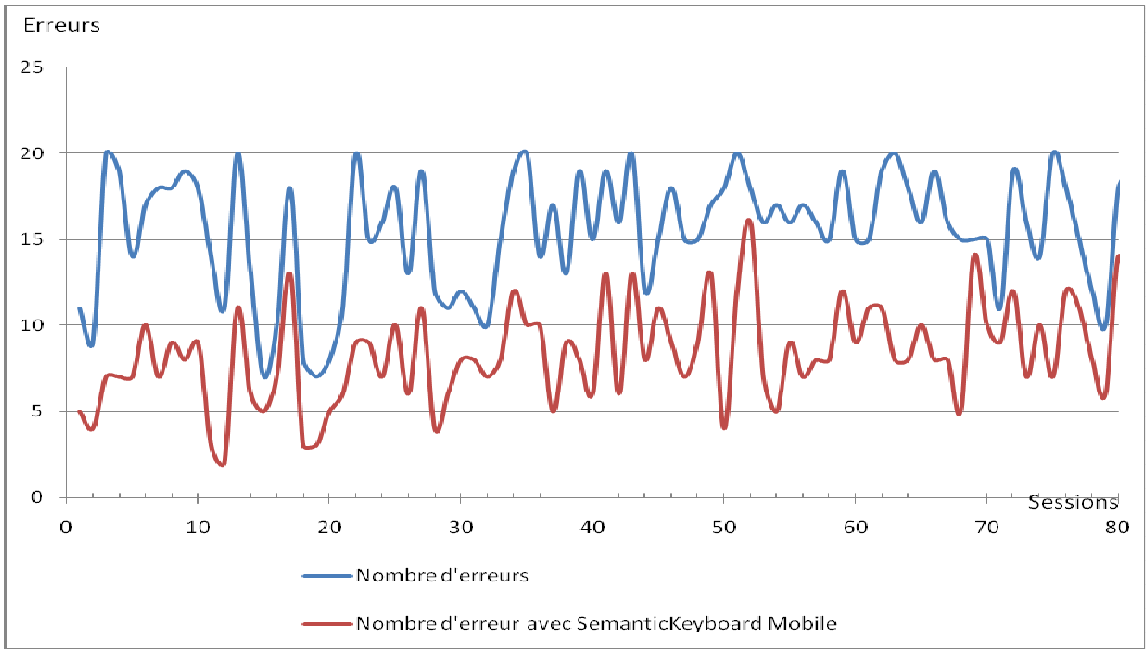


Figure 75. Erreurs avec et sans SemanticKeyboard Mobile pour le premier utilisateur

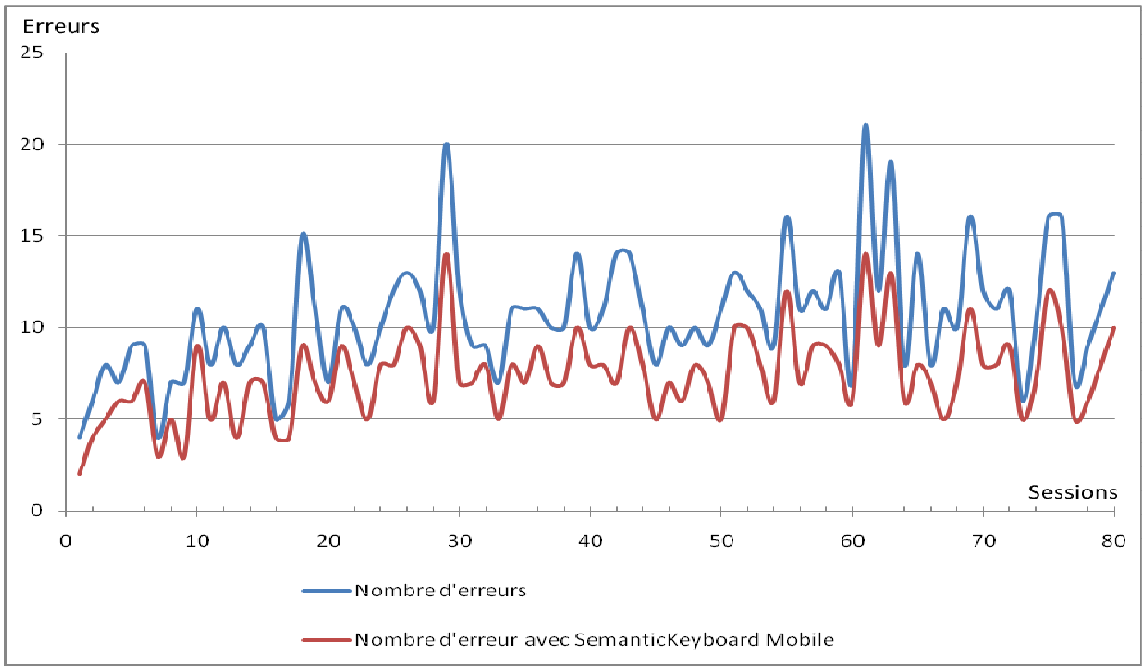


Figure 76. Erreurs avec et sans SemanticKeyboard Mobile pour le second utilisateur

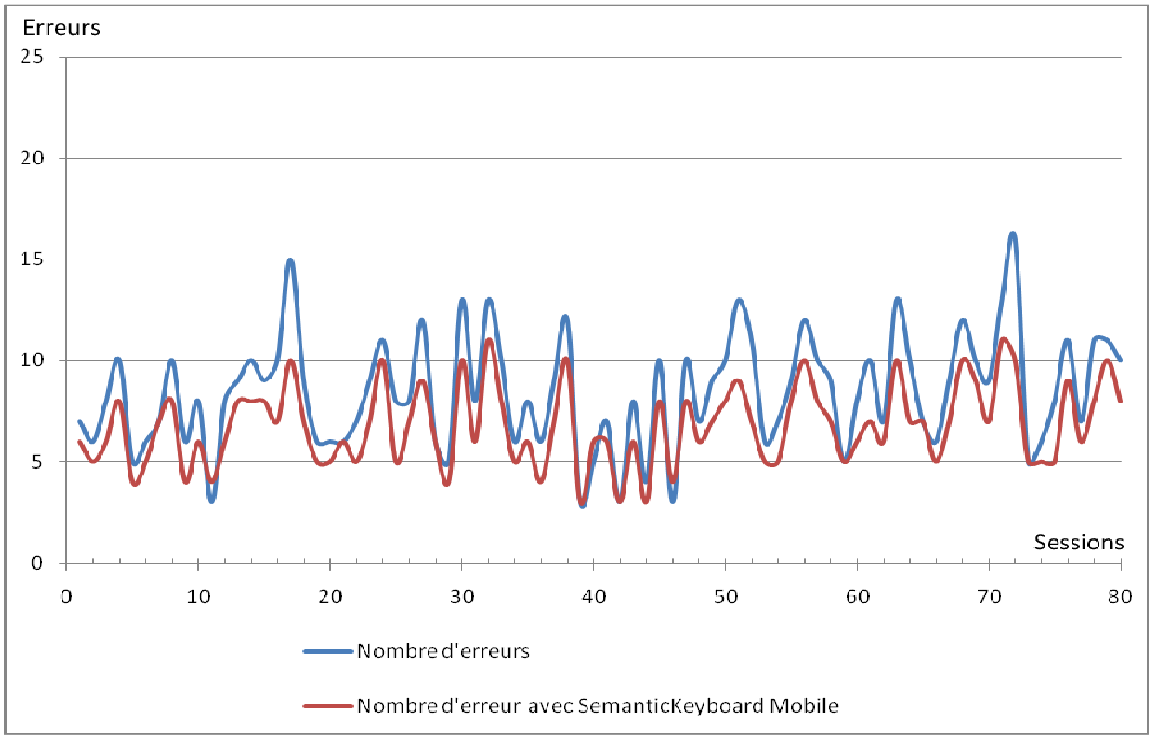


Figure 77. Erreurs avec et sans SemanticKeyboard Mobile pour le troisième utilisateur

Nous observons des résultats extrêmement variables entre les différents utilisateurs. Comme nous pouvons le mesurer Figure 75, pour le premier utilisateur, le nombre d'erreurs a été réduit très significativement (environ 50%) par le système de SemanticKeyboard Mobile. Pour le second utilisateur (cf. Figure 76), en revanche, le bénéfice apporté par le système s'est avéré très limité. Le système se serait même comporté de manière contreproductive pour certaines sessions en générant plus d'erreurs qu'il n'aurait été capable d'en corriger. Les résultats pour le troisième utilisateur (cf. Figure 77) sont un moyen terme entre les deux situations.

Sans grande surprise, on observe que plus l'utilisateur fait de fautes, plus le système semble s'avérer efficace. La nature des erreurs corrigées par le système – erreurs d'imprécision dans le pointage – sont moins nombreuses pour un utilisateur « plus appliqué ». Cette observation confirme que le système serait sans doute plus adapté pour des dispositifs mobiles, dont la tâche est compliquée par un espace réduit et des petites imprécisions dues au mouvement, que pour un environnement bureautique moins restreignant en terme de taille des touches.

Chapitre 12 - Clavier multi-layer

Basé sur l'idée qu'un clavier ZAERTY est déjà plus efficace qu'un clavier AZERTY et dans le but d'éviter une dégradation trop importante des performances au cours des premiers usages, Bi [Bi 10] propose le « clavier Quasi-QWERTY » : le clavier est muni d'une nouvelle distribution de caractères partiellement optimisée. Cependant la stratégie d'optimisation doit permettre de faciliter la recherche visuelle des caractères au cours des premiers usages. Les modifications opérées sur le clavier (QWERTY) sont guidées par l'heuristique suivante : les caractères doivent être placés à leur position d'origine ou adjacents (par les côtés ou en diagonale) à leur position d'origine sur le clavier (cf. Figure 78).

q	w	d	r	t	u	y	l	k	p
z	a	s	e	h	n	i	o	m	
	x	f	v	c	g	b	j		

q	w	e	r	t	y	u	i	o	p
a	s	d	f	g	h	j	k	l	
	z	x	c	v	b	n	m		

Figure 78. Comparaison des claviers Quasi-QWERTY (gauche) et QWERTY (droite) [Bi 10]

L'idée du clavier quasi-QWERTY repose sur le fait que l'usage d'un clavier virtuel nécessite normalement un contrôle visuel[61], à la différence d'un clavier physique qui peut facilement être utilisé en aveugle avec un peu d'expérience. En conséquence, l'utilisateur doit facilement retrouver un caractère si celui-ci se trouve dans le voisinage de l'endroit ciblé visuellement. Les évaluations montrent effectivement que la perte d'efficacité au cours des premiers usages est minimisée par rapport à un clavier optimisé librement (tel que ATOMIK). Cette perte reste néanmoins significative par rapport au clavier QWERTY[62].

Le clavier multi-layer est motivé par le même objectif de fond, à savoir un compromis entre une optimisation de la distribution des caractères moins efficace qu'une distribution optimisée librement mais en compensation moins perturbante pour un utilisateur novice. Il est en revanche inspiré par le concept d'interfaces multi-layer proposé par Ben Shneiderman [Shneiderman 03, Kang 03] et revisité au cours de travaux que nous avons conduit antérieurement [Merlin 07, Merlin 08, Merlin 10c].

Section I - Concept d'interfaces multi-layer

Les interfaces multi-layers sont initialement destinées à permettre à une population hétérogène aux objectifs et niveaux de formations hétérogènes (novice, amateur, expert…) d'utiliser efficacement une interface. Ces interfaces proposent de discriminer différents types d'usages (déclinés du plus superficiel au plus professionnel) et d'activer des fonctionnalités [Shneiderman 03] ou de raffiner l'usage des fonctionnalités [Clark 05] et d'accroître la charge du système de visualisation [Christiernin 05] relativement au niveau d'utilisation.

Le groupe de fonctionnalités et d'éléments de représentation offerts à l'utilisateur définit un layer. Le layer actif ainsi que la transition entre deux layers sont soit contrôlés par l'utilisateur, soit auto déterminés par le système en fonction de l'analyse de l'utilisation faite

[61] Même si certains claviers tels que BlindType (http://www.youtube.com/watch?v=M9b8NIMd79w) sont élaborés dans le but d'un usage aveugle

[62] Temps de saisie au cours des premiers usages de 2110ms pour QWERTY, 3234ms pour Quasi-QWERTY et 4705 pour ATOMIK

par l'utilisateur du logiciel [Clark 05]. Cette auto détermination pourrait rapprocher la notion d'interface multi-layer des interfaces adaptant l'outil au contexte, comme dans le projet Svalabard [Huot 04] et des interfaces suggestives [Igarashi 01].

L'objectif induit du principe d'interface multi-layer est de conduire l'utilisateur à évoluer progressivement dans son usage de l'application tout en conservant continuellement la maîtrise de celle-ci.

Au cours des travaux menés dans le cadre du projet ASTER [Benhacène 05, Benhacène 07, Merlin 10c], nous avons proposé d'adapter le concept d'interface multi-layer pour, cette fois-ci, inciter une population homogène à faire évoluer ses méthodes de travail [Merlin 07, Merlin 08]. Cette démarche avait non seulement pour objectif d'accompagner cette population dans la transition d'anciennes méthodes de travail vers de nouvelles méthodes de travail, mais elle permettait également de casser une barrière psychologique. Au lieu de provoquer une rupture brusque et globale perturbant l'utilisateur et générant des réticences au changement, elle l'engageait à accepter plusieurs petits changements successifs, chacun ayant un impact modéré sur ses méthodes de travail. La démarche engageait ainsi l'utilisateur dans un processus permanent de mutation douce de son environnement.

Section II - Principes du clavier multi-layer

L'origine du clavier multi-layer repose sur cette idée d'accompagner l'utilisateur dans la transition de la distribution de touches originelle (AZERTY, QWERTY, etc.) vers une distribution de touches optimisée. La démarche vise à briser la barrière psychologique du layout AZERTY en engageant l'utilisateur à accepter une modification insignifiante de son clavier logiciel, puis une autre, et une autre jusqu'à obtenir finalement une distribution de touches efficace.

Cette transition est basée sur un nombre fini d'étapes. Au cours de chacune d'elles est opérée une permutation de deux caractères dont les touches sont adjacentes par le côté (pas de permutation en diagonale à l'inverse du clavier Quasi-QWERTY de Bi [BI 10]). Cette permutation simple ne doit pas avoir de conséquences significatives sur le temps de saisie (ni positives, ni négatives), en revanche la répétition des permutations doit progressivement apporter un avantage concret.

Cependant, un temps d'utilisation doit être laissé entre chaque étape de manière à ce que l'utilisateur assimile indépendamment chaque mutation. En conséquence, en fonction du temps d'assimilation de l'utilisateur, de la fréquence d'usage de l'artefact et du nombre d'étapes, la transition peut s'avérer longue voire très longue.

Section III - Implémentation

L'implémentation que nous proposons consiste à effectuer, à chaque étape, la permutation la plus avantageuse en termes de gain de temps (la position du caractère espace reste inchangée). La comparaison du gain de temps est effectuée sur la base du calcul du *upper-bound* [Soukoreff 95] pour chaque permutation possible. Les étapes sont poursuivies jusqu'à ce que toutes les permutations possibles engendrent une perte de temps.

Cette stratégie simple nous permet d'atteindre la version finale du clavier (cf. Figure 80) en 39 étapes[63] [64]. Les performances théoriques du clavier évoluent de 28.7wpm (performances du clavier AZERTY) à 38.5wpm[65] au terme des 39 étapes (cf. Figure 79). A noter que cette stratégie ne garantit pas des performances optimales. Par ailleurs, elle ne modifie pas la géométrie globale du clavier et ne permet donc pas d'atteindre des performances tout à fait équivalentes à des claviers tels que ATOMIK [Zhai 00] ou GAG [Raynal 05b] par exemple.

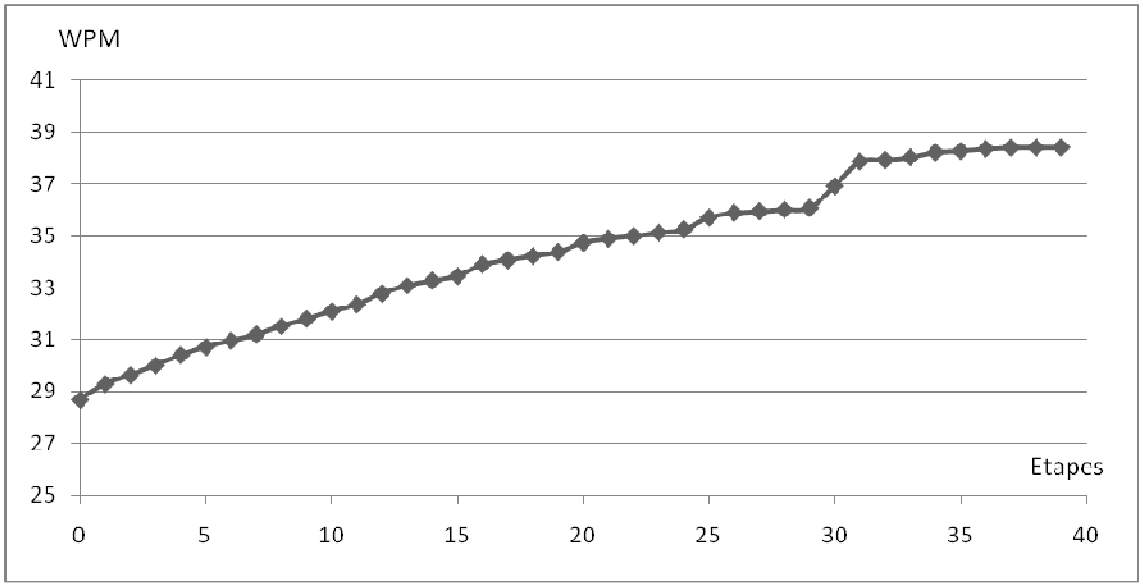


Figure 79. Progression des performances du clavier au cours des étapes

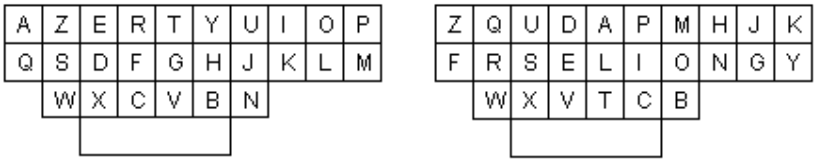


Figure 80. Évolution de la distribution de touches du clavier multi-layer entre la première étape (à gauche) et la dernière étape (à droite)

Section IV - Évaluation

D'un point de vue théorique, l'évaluation laisse envisager un gain en termes de vitesse de saisie d'environ un tiers du temps par rapport au clavier AZERTY. Les distances parcourues par le dispositif de pointage devraient parallèlement être réduites d'environ 30%.

[63] Liste des permutation effectuées à chaque étape : (A;Z) (E;D) (E;F) (S;F) (K;L) (J;L) (Y;U) (Y;I) (H;L) (G;L) (D;R) (A;R) (T;U) (K;M) (Y;O) (Y;P) (B;N) (J;M) (T;G) (G;I) (C;V) (T;N) (I;N) (G;O) (G;P) (H;M) (Q;F) (R;Q) (A;D) (A;U) (D;U) (O;P) (N;P) (N;O) (O;M) (N;H) (C;T) (Y;K) (G;J)

[64] Une démarche similaire d'optimisation du clavier QWERTY pour le portugais nous permet de passer de 27.8wpm à 37wpm au terme des 36 étapes d'optimisation suivantes : (A;S) (A;D) (A;F) (D;F) (S;F) (Y;U) (Y;I) (Y;O) (Y;P) (O;K) (J;O) (K;P) (H;O) (G;O) (B;N) (B;M) (A;O) (V;N) (V;M) (A;N) (Q;W) (J;L) (I;H) (H;P) (G;I) (U;P) (T;P) (R;P) (G;L) (H;G) (H;B) (Z;X) (N;I) (I;A) (C;I) (K;J)

[65] Valeurs théoriques obtenues avec comme coefficients pour la loi de Fitts les paramètres utilisés par Soukoreff et Mackenzie [Soukoreff 95]

Du point de vue de l'évaluation heuristique, le clavier semble plus adapté pour un usager régulier et devrait notamment s'avérer particulièrement efficient pour des usagers handicapés moteurs.

De nombreux paramètres du clavier, et notamment le temps d'utilisation nécessaire à l'assimilation complète d'une permutation, les conséquences du non usage du clavier durant un certain temps sur le « désapprentissage » des permutations antérieures, etc., restent néanmoins difficiles à appréhender du point de vue théorique comme du point de vue heuristique.

Au cours d'une première expérimentation, nous avons choisi d'évaluer le clavier dans un contexte qui devrait s'avérer relativement favorable : utilisations par des utilisateurs normaux mais de façon régulière et peu espacée dans le temps.

IV.1. Protocole expérimental

IV.1.a. Participants

Nous avons réalisé quatre études de cas auprès de quatre utilisateurs volontaires : deux brésiliens et deux français. Les utilisateurs étaient des usagers quotidiens de l'ordinateur et occasionnels de claviers logiciels âgés de 22 à 33 ans.

IV.1.b. Matériel utilisé

Afin de pouvoir mesurer la distance parcourue par le pointeur, la tâche a été réalisée dans un environnement bureautique via la plateforme E-Assist II et les touches du clavier étaient actionnées au moyen du pointeur de la souris. Les touches avaient pour dimension 20*20pxs soit environ 6*6mm sur l'écran utilisé au cours de l'expérimentation (à l'exception de la touche espace dont les dimensions étaient de 60*20pxs).

IV.1.c. Tâche et stimulii

La tâche consistait à recopier, au moyen du clavier logiciel proposé, des phrases courtes affichées à l'écran. Les phrases à recopier ont été construites sur la base de l'étude de la fréquence d'apparition des caractères et des bigrammes de la langue utilisée et de manière à être représentatives de la langue maternelle de l'utilisateur.

Le texte à saisir était présenté dans un bandeau situé au-dessus du texte déjà saisi (cf. Figure 61). L'exercice était initié par une pression sur la touche espace. Une pression sur la touche espace permettait de passer à la phrase suivante autorisant l'utilisateur à effectuer une courte pause, non comptabilisée dans le temps de saisie, entre deux phrases.

Les erreurs n'étaient ni affichées ni corrigées, mais signalées par un feedback sonore et visuel. Le bandeau contenant les caractères saisis restait inchangé jusqu'à ce que l'utilisateur presse le caractère correct.

IV.1.d. Design

Les premières sessions ont été effectuées au moyen du clavier AZERTY. L'exercice a été répété durant 10 sessions au terme desquelles a été observée une stabilisation des performances pour les quatre utilisateurs (illustrant que l'utilisateur dominait complètement l'exercice : complètement le clavier AZERTY ainsi que les phrases de l'exercice). Cette démarche avait pour objectif de mesurer ensuite uniquement les gains (vs pertes) de temps relatifs aux permutations : chacune devait permettre d'améliorer les performances d'un point de vue mécanique, mais pouvait impacter les performances du point de vue cognitif en nécessitant potentiellement un temps supplémentaire de recherche des caractères.

Suite à cette phase d'apprentissage, les utilisateurs étaient amenés à répéter le même exercice de recopie de 10 phrases courtes, à un rythme de 5 sessions par jour, espacées d'au moins 20 minutes. De manière à réduire le temps de l'expérimentation, le nombre des permutations a été réduit à 30. Les permutations des caractères étaient effectuées au fil des sessions à un rythme fixé par chaque utilisateur. L'utilisateur sollicitait une permutation lorsqu'il estimait que les permutations antérieures avaient été complètement assimilées.

IV.2. Résultats

Pour des raisons logistiques, l'un des quatre utilisateurs (utilisateur français) n'a pas pu aller au terme de l'expérimentation (expérimentation arrêtée après la 22^{ième} permutation). En outre, la régularité des sessions n'a pas pu être respectée pour cet utilisateur. Plusieurs fois au cours de l'expérimentation, l'utilisateur a dû interrompre une ou deux journées le déroulement des sessions. Les résultats partiels pour cet utilisateur confirment néanmoins les tendances observées pour les trois autres utilisateurs.

Les résultats présentés ci-dessous (cf. Figure 81, Figure 82 et Figure 83) confirment les hypothèses que nous avions formulées. En effet, les utilisateurs n'ont pas ressenti de difficulté particulière à s'adapter aux mutations successives du clavier. En conséquence, ils ont bénéficié pleinement des améliorations progressives, du point de vue mécanique, apportées par les permutations, soit un gain moyen de 10 mots par minute.

La tendance observée sur l'ensemble des utilisateurs était d'accepter plusieurs permutations successives (deux ou trois) dans des zones distinctes du clavier, puis d'effectuer un palier de trois ou quatre sessions de manière à intérioriser complètement ces mutations.

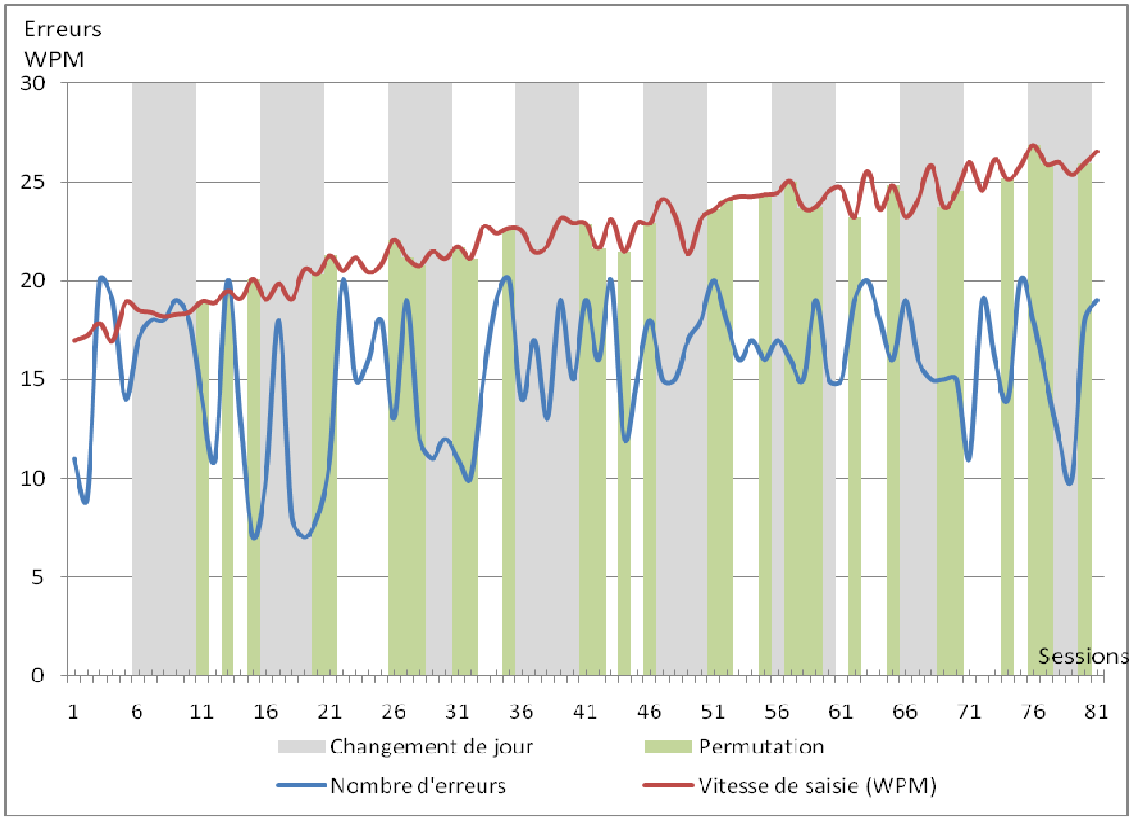


Figure 81. Evolution des performances au cours des sessions pour le premier utilisateur

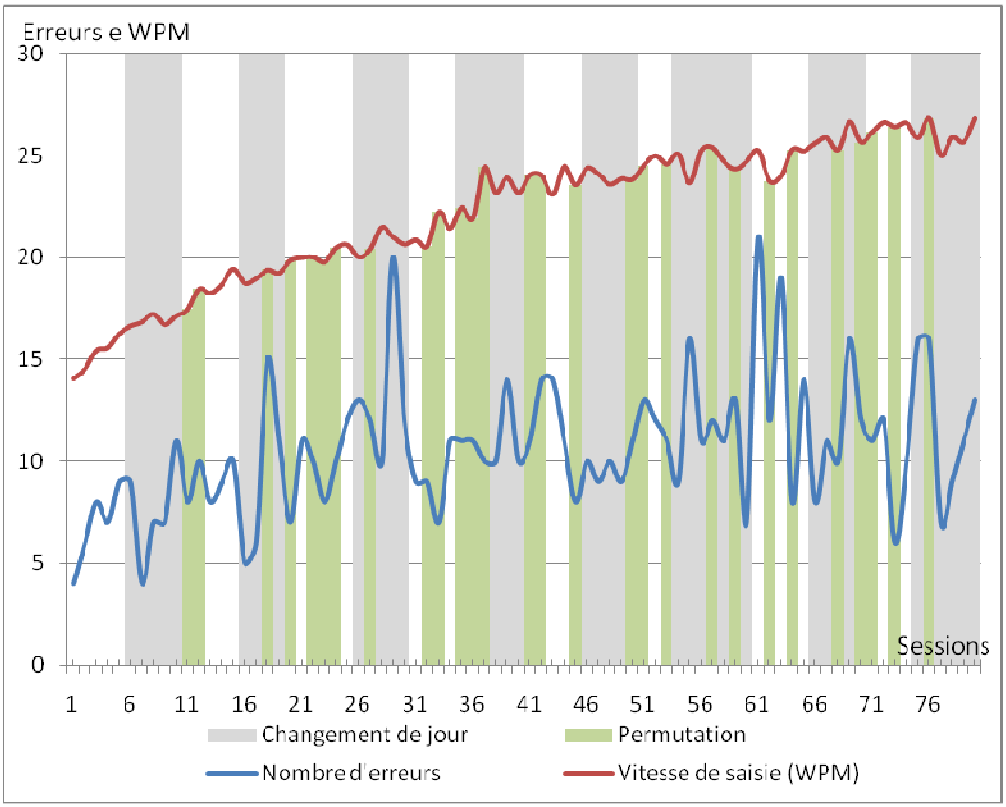


Figure 82. Evolution des performances au cours des sessions pour le second utilisateur

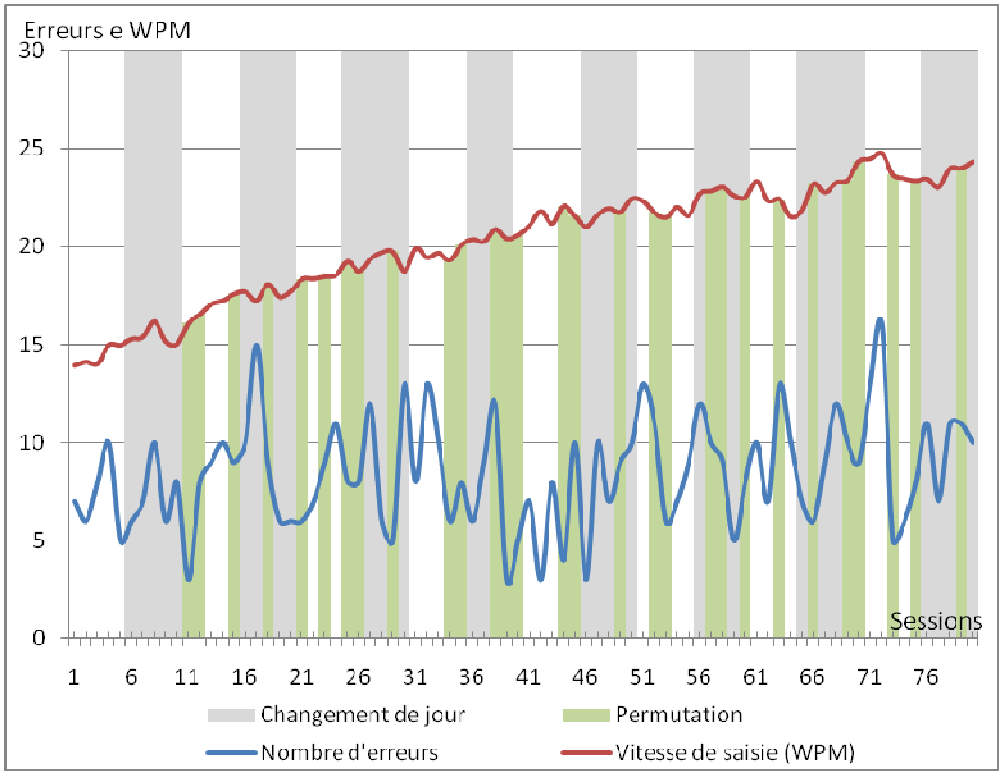


Figure 83. Evolution des performances au cours des sessions pour le troisième utilisateur

L'analyse des erreurs (cf. Figure 81, Figure 82 et Figure 83) n'a pas apporté d'informations significatives. Pour l'ensemble des utilisateurs, le nombre d'erreurs s'est avéré relativement élevé et très irrégulier au cours des différentes sessions. Les variations n'ont pas pu être imputées à des circonstances particulières de l'expérimentation telles que permutation ou changement de jour. On observe néanmoins une légère tendance à un nombre d'erreurs supérieur pour les utilisateurs les plus rapides. Les utilisateurs ont également souligné la relative difficulté de la tâche en raison de la très petite taille des touches. Cette tâche était donc particulièrement sensible aux détails de positionnement ergonomique de la main, aux légers tressautements de la souris, etc., ayant une conséquence ressentie sur le nombre d'erreurs.

IV.3. Conclusion

Le clavier a donc permis un gain de temps très significatif sans l'exigence d'une phase à proprement parler d'apprentissage dans la mesure où l'utilisateur reste continuellement « expert » avec le clavier utilisé. La transition progressive, structurée sur la base du concept d'interface multi-layer, a joué efficacement son rôle d'accompagner l'utilisateur dans une évolution continue de son environnement de travail, sans que cette transition n'ait un coût cognitif ou psychologique fort. Au contraire, les utilisateurs ont perçu l'intérêt des mutations et sont devenus demandeurs de celles-ci.

Néanmoins, cette transition s'est effectuée au cours d'un usage relativement intensif du clavier (environ 20 minutes par jour). Cet usage correspond beaucoup plus à l'usage d'un clavier virtuel par un utilisateur handicapé moteur que par des utilisateurs lambda saisissant en moyenne quelques SMS par mois. Il est donc difficile de préjuger comment cette phase de transition serait appréhendée sur une période beaucoup plus longue où l'utilisateur serait parallèlement confronté à des claviers AZERTY/QWERTY et avec éventuellement de longues périodes sans un usage du clavier.

Le clavier semble donc particulièrement adapté pour des utilisateurs handicapés moteur des membres supérieurs, usant exclusivement des claviers virtuels via un dispositif de pointage. Il conviendrait également idéalement pour quelques autres utilisateurs faisant un usage quotidien du clavier. Il est impossible de conclure sur le long terme pour des utilisateurs lambda. Cependant, il est bon de rappeler que la première permutation (clavier ZAERTY) engendre un clavier qui est déjà plus performant que le clavier AZERTY. Faire accepter définitivement cette simple permutation serait donc déjà un progrès substantiel dans la perspective d'une étape future.

Enfin, comme nous l'avons préalablement évoqué, la stratégie de permutation mise en œuvre ne garantit ni une distribution finale optimale des touches ni un nombre minimal de permutations. Fort des résultats positifs démontrés, reste à définir une stratégie offrant les meilleurs résultats en acceptant ponctuellement quelques régressions opportunes des performances au cours des permutations.

Conclusion et perspectives

Dès lors que l'on s'intéresse à l'optimisation de la saisie sur la base des 26 caractères de l'alphabet latin, on peut observer que, bien que les travaux autour du clavier AZERTY visant à améliorer les propriétés mécaniques du clavier sans affecter la distribution des caractères puissent apporter des bénéfices légers (cf. SemanticKeyboard et SemanticKeyboard Mobile), la réorganisation statique de la distribution des caractères reste l'outil le plus efficace pour obtenir un gain important de performances à long terme.

Dans cette perspective, le clavier multi-layer apparaît comme très bien adapté dans la mesure où il permet de faire progresser un utilisateur vers une nouvelle distribution de caractères tout en démystifiant les changements opérés. Il se révèle ainsi efficace du point de vue des outils d'évaluation traditionnels, dans la mesure où il permet un gain de temps substantiel pour un utilisateur expert, mais il est également performant du point de vue de l'évaluation heuristique car il permet une prise en main directe par les utilisateurs habitués au clavier AZERTY[66].

Néanmoins, l'évaluation heuristique fait émerger que d'autres gains sont envisageables dès lors que l'on s'intéresse plus attentivement à la saisie de texte dans un cadre d'utilisation réelle. Ainsi, les buts de l'utilisateur en termes de formatage sont trop souvent occultés.

Ainsi, nous avons pu constater que beaucoup d'efforts relatifs à l'amélioration des claviers logiciels se concentrent sur les caractères principaux (soit les 26 caractères de l'alphabet latin et le caractère espace). Bien que des gains importants soient théoriquement envisageables en travaillant sur la distribution de ces caractères [Zhai 02a, Raynal 05b, Lesher 00], nous pouvons observer que, dans un cadre d'utilisation réel, les distributions de touches telles qu'AZERTY pour les claviers simples et alphabétiques pour les claviers ambigus, restent des références quasi immuables.

Parallèlement, pour les utilisateurs quelconques, les améliorations proposées autour des claviers AZERTY n'ont pas toujours un impact très significatif.

Pourtant, comme nous l'avons évoqué en première partie, les objectifs de l'utilisateur en matière de saisie impliquent, au-delà de l'entrée de ces 27 caractères, la saisie de caractères de ponctuation, de numéros, de caractères accentués (riches dans la langue française et plus encore en portugais[67]), la discrimination des majuscules et minuscules, etc. Par ailleurs, outre l'espace, certaines touches telles que la délétion ou le retour chariot requièrent un accès privilégié.

L'hypothèse que nous souhaitons formuler en perspective, relativement aux critères heuristiques mis en évidence, est que des gains importants sont potentiellement accessibles au-delà de la saisie des caractères principaux.

Prenons par exemple le cas assez défavorable de la langue portugaise. Les caractères accentués particulièrement diversifiés et fréquents sont saisis par l'intermédiaire de la sélection de l'accent (accent circonflexe, accent aigu, accent grave ou tilde) puis de la lettre accentuée (cf. Figure 84). En conséquence, sur un clavier virtuel, trois frappes successives sont nécessaires pour saisir le caractère « à » (Shift, puis la touche contenant les deux

[66] Performant pour les populations en question : utilisateurs aguerris au clavier AZERTY

[67] Nous rappelons qu'une partie des travaux relatifs à cette thèse ont été effectués au Brésil

accents grave et aigu, puis la lettre « a ») et plus encore pour la saisie d'une majuscule accentuée. En conséquence, sans qu'aucune erreur de frappe ne soit décomptée, le KSPC atteint une valeur de 1,18 pour la saisie d'un texte simple[68]. Par ailleurs, cette saisie des accents et de la cédille requiert trois touches dédiées réduisant d'autant l'espace affecté aux autres caractères.

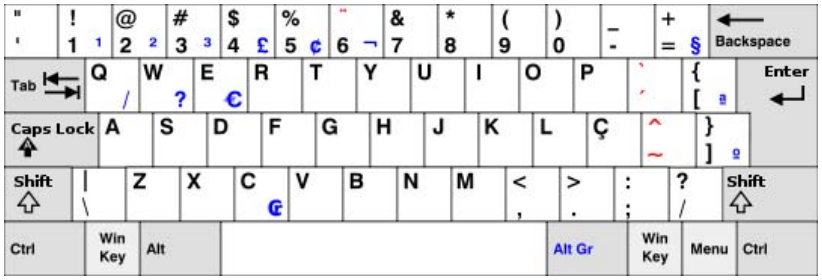


Figure 84. Distribution de touches du clavier portugais-brésilien[69]

Un peu à l'image de ce que propose le clavier de l'iPhone, nous avons proposé de substituer la succession de pressions de touches nécessaire à la saisie des caractères accentués, du c cédille et des majuscules (ou caractère secondaire de la touche), ainsi que l'accès aux touches espace, retour chariot et délétion, par des gestes intuitifs et rappelant le symbole, ou la fonction (cf. Figure 85 et description KeySpec du clavier en annexe B).

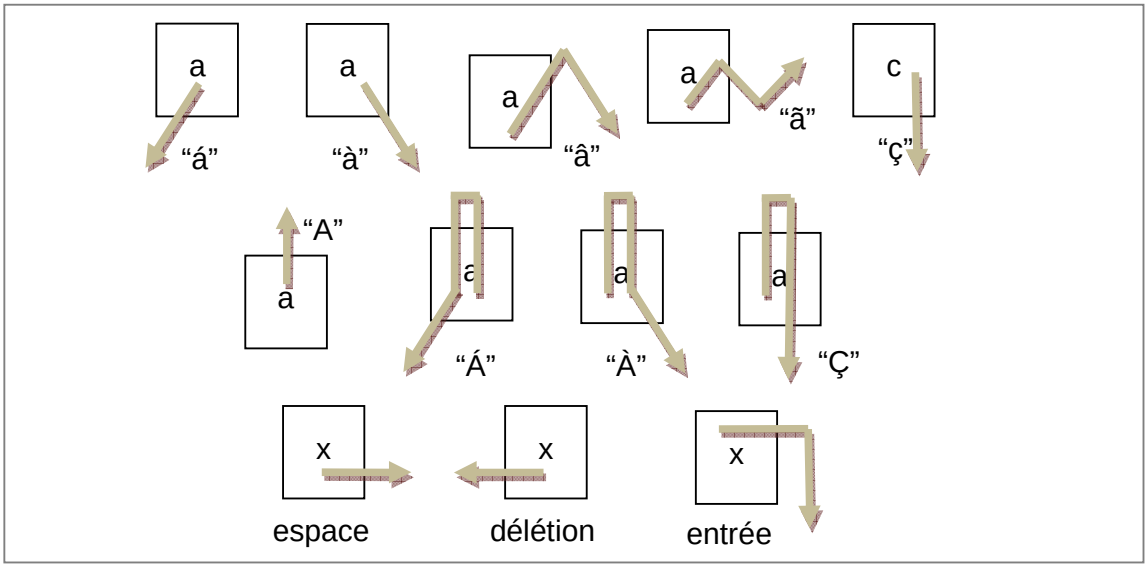


Figure 85. Vocabulaire de gestes

En plus de permettre de rétablir un KSPC de 1 pour le texte préalablement étudié, cette démarche nous permet d'économiser un espace substantiel sur le clavier (cf. Figure 86) tout en accélérant l'accès à certaines touches fréquemment utilisées.

[68] Extraits de la version portugaise du « Petit Prince », en considérant majuscules, accents, et en conservant les caractères de ponctuation virgule, point-virgule, point, deux points, tirets, guillemets, apostrophes, points d'interrogation et d'exclamation.

[69] http://fr.wikipedia.org/wiki/Disposition_des_touches_des_claviers_informatiques

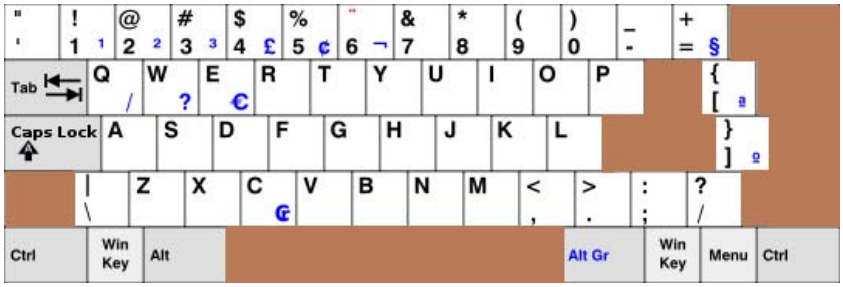


Figure 86. Économie d'espace réalisée (en marron)

Ces premiers travaux préliminaires que nous avons pu effectuer laissent présumer la possibilité d'obtenir des gains tout à fait significatifs sur la saisie en travaillant en marge de la saisie des 26 caractères de l'alphabet latin.

Conclusion et Perspectives

Nous sommes partis de l'hypothèse que les méthodes traditionnelles d'évaluation et de comparaison des claviers logiciels se basaient sur un aspect réducteur, et pas toujours objectif, relatif à leur usage : les performances des utilisateurs experts. En outre, le procédé d'évaluation étant un guide, une référence implicite pour le design des artefacts, nous nous interrogions sur le risque d'un effet pervers sur la nature des artefacts produit résultant de cette base de comparaison. Plus concrètement, en se focalisant sur les performances à long terme, le risque est de produire des artefacts efficients pour des utilisateurs experts mais difficiles à prendre main et consécutivement rejetés par la grande majorité des utilisateurs.

A travers l'étude de ces méthodologies d'évaluation, expérimentales ou basées sur des modèles prédictifs, ainsi qu'à travers les travaux conduits pour améliorer les modèles prédictifs, nous avons effectivement conclu à une absence de significativité des données exploitées à titre de comparaison des claviers, sous couvert pourtant du caractère rationnel et objectif de celles-ci. Nous avons, entre autres, mis en évidence l'ambigüité entre les notions d'utilisateurs « novices » ou « débutants » et « experts » ou « parfaits » compliquant l'établissement d'une référence chiffrée.

Nous en avons déduit que les claviers logiciels ne peuvent être évalués uniquement sur la base d'un critère intrinsèque et décorrélé de leurs utilisateurs, et, plus largement, décorrélé de leurs utilisateurs, relativement à un ou des usages et à un ou des contextes d'utilisation. En réponse, nous avons proposé une grille d'évaluation heuristique permettant de mesurer les caractéristiques des claviers relativement à l'ensemble de ces aspects et de fournir concrètement une évaluation par rapport à une combinatoire de paramètres : utilisateurs types ; usage ; fréquence d'usage ; et contexte d'utilisation.

Cette grille d'évaluation heuristique n'a pas pour finalité de substituer complètement tout autre mode d'évaluation. Mais elle doit, en premier lieu, permettre d'offrir une vision plus nuancée des performances d'un clavier logiciel, et, parallèlement, de proposer d'autres critères à satisfaire que les simples performances à long terme pour qu'un clavier puisse être considéré comme plus performant qu'un autre. Ce deuxième objectif vise implicitement à offrir une base théorique pour influencer les critères de design des claviers logiciels et à orienter explicitement le design de manière à ce qu'un clavier soit en mesure d'être approprié par une population.

En prenant comme référence de comparaison cette évaluation heuristique, sans toutefois ignorer la nécessité, pour les nouveaux claviers, d'améliorer les performances des claviers existants, nous avons identifié des critères discriminants pour obtenir des performances plus satisfaisantes. Sur la base de ces critères nous avons conçu et étudié quatre nouveaux paradigmes : SpreadKey, SemanticKeyboard, le clavier multi-layer et enfin l'amélioration de

l'accès aux caractères « secondaires » du clavier (tels que majuscules, caractères accentués etc.).

Les premiers travaux effectués autour de SpreadKey nous ont permis d'observer que, bien que les caractères soient systématiquement accessibles à la même place que leur place sur le clavier AZERTY, ce qui permettait de répondre de façon satisfaisante à certains critères de l'évaluation heuristique, les changements dynamiques apportés sur le clavier perturbent l'utilisateur. Même s'il a la possibilité de procéder normalement à la saisie, l'utilisateur s'interrompt spontanément pour analyser brièvement les changements. En conséquence, la saisie s'en retrouve ralentie et plus fatigante d'un point de vue cognitif. Au final, seuls les utilisateurs ralentis par un fort handicap moteur ont eu un bénéfice substantiel à utiliser le clavier. Ces travaux nous ont finalement permis d'identifier que la grille d'évaluation heuristique actuelle ne répondait pas de façon totalement satisfaisante à cette question de la dynamicité de l'espace visuel.

Le deuxième axe de travail s'est consécutivement orienté vers une aide dynamique à la saisie, mais cette fois-ci sans modification de l'espace visuel. Ces travaux se sont matérialisés par l'exploration des concepts SemanticKeyboard et SemanticKeyboard Mobile. Respectivement dans un environnement bureautique et sur des supports mobiles, les deux claviers visaient à modifier dynamiquement les propriétés mécaniques des claviers en fonction de la probabilité des caractères à saisir, mais sans impacter l'espace visuel. SemanticKeyboard influait sur la trajectoire du pointeur pour faciliter l'accès aux caractères les plus probables, tandis que SemanticKeyboard Mobile augmentait sensiblement l'espace des touches contenant les caractères les plus probables, sans en modifier l'aspect, de manière à ce qu'une frappe à la limite entre deux touches profite au caractère le plus probable. Les résultats se sont avérés un peu plus satisfaisants, notamment en ce qui concerne la version mobile, qui devrait permettre de corriger de façon significative les erreurs dans des contextes d'interaction dégradées par la mobilité. En revanche, les dispositifs n'influent pas de façon probante sur la vitesse de saisie.

Pour obtenir une amélioration significative de la vitesse de saisie à long terme, les réorganisations statiques de la distribution des touches restent un moyen très efficace. Mais ces réorganisations sont pénalisées par un fort coût d'apprentissage rendant leur acceptabilité difficile. Les difficultés, liées à la recherche des caractères, rencontrées par les utilisateurs au cours des premiers usages dégradent les performances à court terme et découragent l'utilisateur. Inspirés par le concept d'interface multi-layer nous avons, en conséquence, proposé une stratégie pour accompagner l'utilisateur d'une distribution classique de touches vers une nouvelle distribution optimisée. Cette stratégie, matérialisée dans le clavier multi-layer et consistant à effectuer des permutations une à une sur le clavier au cours d'une période transitoire, a permis d'effacer une barrière psychologique relative à l'organisation des touches. Elle a offert des résultats très satisfaisants tant du point de vue de l'acceptation des utilisateurs que du point de vue des performances finales obtenues.

Enfin, observant que les évolutions généralement proposées sur les claviers logiciels (optimisations autour des 26 caractères de l'alphabet latin) s'attaquent essentiellement à un sous-objectif de la saisie de textes (saisie de textes sans formatage et sans ponctuation), nous avons proposé de nous intéresser à l'optimisation des claviers dans le cadre d'un objectif plus adéquat à la réalité de leur utilisation. Nous avons ainsi observé qu'un simple clavier, augmenté d'interactions gestuelles simples et intuitives, permettait de réduire de façon significative le nombre de pointages nécessaires à la saisie de caractères accentués, majuscules, etc., et consécutivement nécessaires à la saisie du texte global. Cette réduction du nombre de pointages doit avoir comme impact d'offrir un gain de temps probable sur la saisie.

De manière à conduire l'ensemble de ses travaux d'évaluation et de design de nouveaux claviers, nous avons proposé, sur les bases initiales de la plate-forme E-Assiste, une nouvelle version de la plate-forme : E-Assist II (et sa version pour dispositifs mobiles TinyEAssist).

Cette nouvelle plate-forme a permis, en premier lieu, d'instrumentaliser les évaluations théoriques, heuristiques, et expérimentales des claviers logiciels (dans des environnements aussi divers que bureautique, web ou mobile), et, en second lieu, de mettre à disposition un langage de conception des claviers logiciels facilitant le design de claviers complexes incluant des comportement dynamiques et relatifs à un système de prédiction.

Les perspectives de ces travaux se dessinent autour des trois principaux axes de contribution.

En premier lieu, nous avons proposé une méthode complémentaire d'évaluation des claviers logiciels sous la forme de critères d'évaluation heuristique. Cette démarche nous a permis de comparer les claviers avec un regard plus complet que la simple comparaison sur la base des performances maximales accessibles à long terme avec le clavier. Néanmoins, d'autres initiatives parallèles [Poirier 07] conduisent, par exemple, à évaluer la charge cognitive relative à l'usage du clavier. Dans nos futurs travaux nous serons ainsi amenés à mesurer de façon plus fine le poids à attribuer aux différents critères, à la fois performances, critères heuristiques et autres critères, de manière à formaliser une méthodologie complète d'évaluation des claviers.

En second lieu, à travers l'observation des critères heuristiques proposés, nous avons identifié différents axes d'amélioration des claviers logiciels. Un axe de travail a abouti à un artefact donnant satisfaction (le clavier muti-layer), d'autres ont, malgré des résultats mitigés, ouvert des perspectives probables d'amélioration, enfin, l'optimisation des claviers « complets » (optimisation étendue à l'ensemble des touches de caractère et de formatage), bien que très partiellement évaluée au cours de cette étude, offre des premiers résultats extrêmement encourageants. Sur la base de ces différents résultats, les perspectives sont diverses. Nous prétendons, dans un premier temps, déployer le clavier multi-layer auprès d'une population d'utilisateurs handicapés moteur de manière à étudier son usage dans un contexte *in situ* et sur une période étendue. Dans un second temps, nous prétendons confirmer ou non l'intérêt des claviers SpreadKey, SemanticKeyboard et SemanticKeyboard Mobile à travers une étude de la deuxième version de SpreadKey, une amélioration des algorithmes de SemanticKeyboard et une étude en mobilité de SemanticKeyboard Mobile. Enfin, les premières réflexions autour des claviers complets ont permis d'ouvrir un chantier important qu'il sera nécessaire d'explorer de façon étendue.

Pour terminer, cette étude nous a conduit à élaborer un langage et une plate-forme d'évaluation des claviers logiciels. Le langage nous a permis de faciliter le design de nouveaux claviers ainsi que leur intégration dans un contexte d'évaluation ou bien au sein d'un système d'exploitation. Néanmoins, le support XML du langage reste encore restrictif à des utilisateurs sensibilisés à la programmation de hauts niveaux. Il reste fermé à des utilisateurs potentiels que seraient des thérapeutes ou des éducateurs accompagnant, par exemple, des enfants handicapés. Une des perspectives de nos travaux, relatifs à ce langage, perspective poussée par des travaux initiaux conduit à cet effet [HA 10], serait ainsi l'élaboration d'un outil graphique de design des claviers accessible à un public plus vaste. De la même manière, la formalisation des protocoles expérimentaux, basée sur une description XML, reste relativement hermétique à un public potentiel que seraient les

ergonomes, et consécutivement pourrait être avantageusement supportée par un outil graphique.

Mais, au-delà de ces caractères d'accessibilité de la plate-forme à différents publics, celle-ci propose différents outils d'évaluation à la fois théorique, expérimentale et heuristique. D'autres outils répondant à l'évaluation d'autres caractéristiques, telles que l'évaluation de la charge cognitive relative à l'usage des claviers, viendront renforcer la plate-forme dans un futur proche. Une des principales perspectives futures sera, consécutivement, de formaliser au sein de la plate-forme un processus général d'exploitation des différents outils d'évaluation pour proposer une évaluation complète et normalisée des claviers logiciels.

Annexes

Annexe A – Exemples de protocole d’expérimentation

Protocole expérimental de SpreadKey

```
<experimentation sessions="5">
	<login>
		<variable label="Age" name="age" style="integer" default="18" />
		<variable label="Sexe" name="sexe" style="select" default="">
			<alternative value="Homme"/>
			<alternative value="Femme"/>
		</variable>
		<variable label="Handicap moteur? "
			name="handicap" style="select" default="" >
			<alternative value="Non"/>
			<alternative value="Oui"/>
		</variable>
	</login>

	<questionary_def name="questionnaire_final">
		<variable label="Sexe" name="sexe" style="select" default="">
			<alternative value="Femme"/>
			<alternative value="Homme"/>
		</variable>
		<variable label="&lt;html&gt;Le dispositif SpreadKey vous a-t-il aidé
			&lt;br&gt;à pointer plus facilement les
			caractères?&lt;/html&gt; " name="commentaire1"
			style="area" default=""/>
	</questionary_def>

	<condition name="entrainement"
		keyboard="azerty_spread_ii.xml" instructions="recommandation-5.txt"/>
	<condition name="spreadkey"
		keyboard="azerty_spread_ii.xml" instructions="recommandation-7.txt"/>
	<condition name="azerty"
		keyboard="azerty_simple.xml" instructions="recommandation-8.txt"/>

	<group name="gp1" handicap="Non"/>
	<group name="gp2" handicap="Non"/>
	<group name="gp3" handicap="Oui"/>
	<group name="gp4" handicap="Oui"/>
```

```
        <session context="gp1:0,2,4,6,8;gp2:1,3,5,7,9;gp3:0,2,4,6,8;gp4:1,3,5,7,9">
                <instruction sessions="0"
                        texts="recommandation-1.txt;recommandation-2.txt;recommandation-
                        2bis.txt;recommandation-3.txt;recommandation-4.txt; "/>
                <exercice sessions="0" condition="entrainement" text="entrainement"/>
                <exercice condition="spreadkey" text="sentences"/>
                <exercice condition="azerty" text="sentences"/>
                <questionary session="9" name="questionnaire_final"/>
                <instruction texts="recommandation-10.txt"/>
        </session>
        <session context="gp2:0,2,4,6,8;gp1:1,3,5,7,9;gp4:0,2,4,6,8;gp3:1,3,5,7,9">
                <instruction sessions="0"
                        texts="recommandation-1.txt;recommandation-2.txt;recommandation-
                        2bis.txt;recommandation-3.txt;recommandation-4.txt; "/>
                <exercice sessions="0" condition="entrainement" text="entrainement"/>
                <exercice condition="azerty" text="sentences"/>
                <exercice condition="spreadkey" text="sentences"/>
                <questionary session="9" name="questionnaire_final"/>
                <instruction texts="recommandation-10.txt"/>
        </session>
</experimentation>
```

Protocole expérimental des expérimentations sur la perception de l’ordre alphabétique

```
<Experimentation sessions="1">
        <login>
                <variable label="Âge" name="age" style="integer" default="18"/>
                <variable label="Sexe" name="sexe" style="select" default="">
                        <alternative value="Femme"/>
                        <alternative value="Homme"/>
                </variable>
        </login>

        <condition name="exo" keyboard="test_alpha_preception.xml"/>
        <group name="gp1"/>

        <session context="gp1:0">
                <instruction texts="debut.txt"/>
                <exercice condition="exo" text="caracteres"
                        presentation="presentation.Aleatory" />
                <instruction texts="fin.txt"/>
        </session>
</Experimentation>
```

Annexe B – Exemples de claviers en KeySpec

DashKey

Cet exemple[70] illustre un clavier contenant des touches ambigües (un caractère central et deux ou trois caractères situés aux coins de la touche). La frappe est désambigüisée par interaction : un simple clic pour la saisie du caractère central, un tracé en direction du caractère pour la saisie des caractères situés aux angles.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<keyboardmodel name="dashkey" width="268" height="190">

        <!-- Pour désambigüiser les frappes, nous devons filtrer différentes
             interactions : un simple relâchement de la souris, et des gestes
             directionnels. La direction des gestes est spécifiée par un angle minimum
             (anglemin) et un angle maximum (anglemax). Pour que le déplacement de la
             souris soit considéré comme un geste et non comme un simple pointage, ce
             déplacement doit être supérieur au seuil 'threshold' (exprimé en pixels).
             Lorsqu'un geste en direction de l'angle supérieur droit est détecté,
             l'événement interne 'dash-NO' est notifié ; -->
        <interaction type="dash" event="dash-NO"
                anglemin="270" anglemax="0" threshold="10"/>
        <!-- Respectivement, lorsqu'un geste en direction de l'angle supérieur gauche,
             inférieur droit, inférieur gauche, ou un simple relâchement de la souris
             est détecté, un événement interne 'dash-NE', 'dash-SO', 'dash-SE' ou
             'released' est notifié -->
        <interaction type="dash" event="dash-NE" anglemin="0"
                anglemax="90" threshold="10"/>
        <interaction type="dash" event="dash-SE" anglemin="90"
                anglemax="180" threshold="10"/>
        <interaction type="dash" event="dash-SO" anglemin="180"
                anglemax="270" threshold="10"/>
        <!-- La capture d'un simple clic génère l'évènement interne 'released' -->
        <interaction type="released" event="released"/>

        <!-- Le modèle de touche utilise la représentation de touches par défaut avec
             une taille de 60x60 pixels. Les touches exploitent 5 des 9 caractères
             contenus dans la représentation par défaut. La saisie de ces caractères est
             associée à la réception des évènements 'released', 'dash-NO', 'dash-NE'
             etc. -->
        <keymodel name="key" width="60" height="60">
```

[70] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

```
                <character name="main" svg_suffixe="CENTER" onevent="released"/>
                <character name="NO" svg_suffixe="NO" onevent="dash-NO" />
                <character name="NE" svg_suffixe="NE" onevent="dash-NE" />
                <character name="SE" svg_suffixe="SE" onevent="dash-SE" />
                <character name="SO" svg_suffixe="SO" onevent="dash-SO" />
            </keymodel>

            <!-- La première touche est positionnée de manière absolue aux coordonnées
                (5,5). Le premier caractère, le caractère central est configuré avec la
                valeur 'A',  le second (NO situé dans l'angle supérieur droit) est
                configuré avec la valeur 'B', le troisième (NE) avec la valeur 'C' et les
                deux caractères restant sont dépourvus de valeur -->
            <key type="key" x="5" y="5" characters="A;B;C;;;"/>
            <!-- Les autres touches sont positionnées relativement à la touche qui les
                précède au moyen des variables PREVIOUS_KEY_X, PREVIOUS_KEY_Y,
                PREVIOUS_KEY_H et PREVIOUS_KEY_W qui représentent la position et les
                dimensions de la touche précédemment déclarée -->
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="E;D;F;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="I;G;H;J;;"/>
            <key type="key" x="5"
                            y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H + 1" characters="L;K;;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="M;N;;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="O;P;;;;"/>
            <key type="key" x="5"
                            y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H + 1" characters="R;Q;;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="S;T;;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters="U;V;W;;;"/>
            <key type="key" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W + 1"
                            y="PREVIOUS_KEY_Y" characters=" ;X;Y;Z;;"/>

        </keyboardmodel>
```

KeyGlass

Cet exemple illustre un clavier de type AZERTY pour lequel quatre touches additionnelles, contenant les quatre caractères les plus probables, sont proposées aux angles de la dernière touche frappée : le système KeyGlass[71]. Ces touches additionnelles sont donc

[71] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

positionnées de façon dynamique, et leur contenu évolue également de manière dynamique en fonction des résultats d'un système de prédiction. Les touches sont visibles uniquement si le précédent caractère saisi n'est pas un espace. Dans le cas contraire elles sont masquées.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<keyboardmodel name="keyglass" width="400" height="200">

        <!-- Nous déclarons en premier lieu un système de prédiction (de nom ngram).
             Par défaut, l'algorithme de prédiction prédit le prochain caractère en
             fonction de tous les caractères préalablement saisis. -->
        <prediction name="ngram" dictionary="francais.txt" />

        <!-- Une variable BORDER va nous permettre d'ajuster la position des touches
             par rapport au bord du clavier -->
        <variable name="BORDER" min="0" max="100" step="1" default="25"/>

        <!-- Nous filtrons les interactions de type relâchement de la souris.
             Lorsque la souris est relâchée, un évènement interne "souris_relachee" est
             notifié. -->
        <interaction type="released" event="souris_relachee"/>

        <!-- Un premier modèle de touche permet de décrire les touches standards du
             clavier. Les états graphiques du modèle n'étant pas précisés, la
             représentation graphique par défaut est utilisée. -->
        <keymodel name="normalkey" width="31" height="31">
            <!-- La touche contient un caractère positionné au centre de la touche.
                 Le caractère est saisi lorsque la touche détient le focus et que
                 l'évènement interne "souris_relachee" est émis -->
            <character name="main" svg_suffixe="CENTER" onevent="souris_relachee" />
        </keymodel>

        <!-- Un second modèle de touche permet de décrire les keyglass. Les états
             graphiques sont décrits dans le fichier 'keyglass.svg'. -->
        <keymodel name="keyglass_key"
            pressed="KeyGlass.svg#pressed"
            focused="KeyGlass.svg#idle"
            over="KeyGlass.svg#over"
            released="KeyGlass.svg#idle" width="25" height="25">
            <character name="main" svg_suffixe="char" onevent="souris_relachee" />
        </keymodel>

        <!-- Nous instancions les touches standards du clavier. Ces touches
             correspondent au type spécifié par le keymodel 'normalkey'. Elles sont
             positionnées relativement à la variable BORDER et l'unique caractère de
             chaque touche est initialisé par un triplet (x,x,caractere) où x est le
             caractère représenté sur la touche et le caractère saisi sur réception de
```

```
     l'évènement 'souris_relachee', et 'caractere' un évènement interne notifié
     simultanément sur réception de l'évènement 'souris_relachee'. -->
<key type="normalkey" x="5+BORDER" y="5+BORDER"
       characters="(A,A,caractere)"/>
<key type="normalkey" x="37+BORDER" y="5+BORDER"
       characters="(Z,Z,caractere)"/>
[…]
<key type="normalkey" x="165+BORDER" y="69+BORDER"
       characters="(R,R,caractere)"/>
<key type="normalkey" x="197+BORDER" y="69+BORDER"
       characters="(N,N,caractere)"/>

<!-- La touche espace est identique aux autres touches mais l´évènement interne
     'espace' est notifié en lieu et place de 'caractere' sur réception de
     l'évènement 'souris_relachee'. -->
<key type="normalkey" x="50+BORDER" y="101+BORDER"
       characters="( , ,espace)" w="130"/>

<!-- Le clavier possède deux états internes possibles 'keyglass_invisible' et
     'keyglass_visible'. Initialement, il est dans l'état 'keyglass_invisible'.
     Sur réception d'un évènement interne 'caractere', l'état devient
     'keyglass_visible'. Puis, sur réception d'autres évènements 'caractere',
     l'état reste inchangé, en revanche, sur réception d'un évènement 'espace',
     l'état redevient 'keyglass_invisible'. -->
<fsm default="keyglass_invisible">
       <transition from="keyglass_visible"
              to="keyglass_invisible" onevent="espace"/>
       <transition from="keyglass_invisible"
              to="keyglass_visible" onevent="caractere"/>
</fsm>

<!-- Nous déclarons ensuite les keyglass qui implémentent le keymodel
     'keyglass_key'. Ces touches sont visibles uniquement lorsque le système
     est dans l'état 'keyglass_visible'. Les touches sont positionnées
     relativement à la dernière touche pressée. Le caractère est initialisé en
     fonction du résultat du système de prédiction. -->
<key visible_states="keyglass_visible" type="keyglass_key"
       x="SELECTED_KEY_X - 20"
       y="SELECTED_KEY_Y - 20"
       characters="#PRED(ngram,0)"/>
<key visible_states="keyglass_visible" type="keyglass_key"
       x="SELECTED_KEY_X + SELECTED_KEY_W" y="SELECTED_KEY_Y - 20"
       characters="#PRED(ngram,1)"/>
<key visible_states="keyglass_visible" type="keyglass_key"
       x="SELECTED_KEY_X - 20" y="SELECTED_KEY_Y + SELECTED_KEY_H"
       characters="#PRED(ngram,2)"/>
```

```
        <key visible_states="keyglass_visible" type="keyglass_key"
               x="SELECTED_KEY_X + SELECTED_KEY_W" y="SELECTED_KEY_Y + SELECTED_KEY_H"
               characters="#PRED(ngram,3)"/>
        </keyboardmodel>
```

TouchPal

Cet exemple illustre un autre usage d'un système de prédiction. TouchPal[72] propose deux modes d'utilisation des touches. Les touches ambigües de TouchPal contiennent deux caractères accessibles par un geste (un trait à droite saisit le caractère de droite, un trait à gauche, le caractère de gauche). Une simple pression de la touche saisit le caractère le plus probable des deux caractères présents sur la touche. Ce fonctionnement nécessite la modification du comportement par défaut des touches via l'addition de code Java.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<keyboardmodel name="dashkey" width="180" height="106">

        <!-- Nous déclarons en premier lieu un système de prédiction (de nom ngram).
             Par défaut, l'algorithme de prédiction prédit le prochain caractère en
             fonction de tous les caractères préalablement saisis. -->
        <prediction name="ngram" dictionary="francais.txt" />
        <!-- Les trois interactions filtrées vont nous permettre de faire la
             discrimination entre une simple pression (notifiant l'événement interne
             'released'), un trait vers la gauche (notifiant l'événement dash-O) et un
             trait vers la droite (notifiant l'événement dash-E) -->
        <interaction type="dash" event="dash-E"
               anglemin="45" anglemax="135" threshold="10"/>
        <interaction type="dash" event="dash-O"
               anglemin="225" anglemax="315" threshold="10"/>
        <interaction type="released" event="released"/>

        <!-- La touche contient trois caractères : les deux caractères "normaux" de la
             touche accessibles par un trait à gauche ou un trait à droite ; et un
             caractère central qui correspondra au caractère le plus probable des deux.
             Ce caractère central est invisible et adapté dynamiquement en fonction du
             contexte. Il est saisi par un simple clic sur la touche -->
        <keymodel name="key" width="31" height="31">
               <character name="main" svg_suffixe="CENTER" onevent="released"/>
               <character name="O" svg_suffixe="O" onevent="dash-O"/>
               <character name="E" svg_suffixe="E" onevent="dash-E"/>
               <!-- Dans le cas de KeyGlass, le contenu des keyglass pouvait être
                    spécifié de façon statique par la fonction #PRED(ngram,X) dans la
                    mesure où la valeur du caractère correspondait systématiquement au
                    Xème résultat du système de prédiction. Dans le cas présent il est
```

[72] Version exécutable accessible à l'adresse http://www.irit.fr/~Bruno.Merlin/research/demo

```
                    nécessaire de spécifier la valeur de manière algorithmique en
                    redéfinissant le comportement de la méthode contextUpdated ()
                    appelée une fois que le système de prédiction a été mis à jour
                    après la précédente saisie. -->
                <function><![CDATA[
                   public void contextUpdated () {
                      super.contextUpdated ();
                      EAssistCharacter carC = getCharacterByName ("main");
                      String carO = getCharacterByName ("O").getDefaultValue ();
                      String carE = getCharacterByName ("E").getDefaultValue ();
                      if (carE.equals ("") ||
                          getProba ("ngram", carO) > getProba ("ngram", carE)) {
                          carC.setCurrentValue ("(," + carO + ",)");
                      }
                      else {
                          carC.setCurrentValue ("(," + carE + ",)");
                      }
                   }
                ]]></function>
        </keymodel>

        <!-- Nous déclarons ensuite les instances des touches -->
        <key type="key" x="5" y="5" characters=";Q;W"/>
        <key type="key" x="37" y="5" characters=";E;R"/>
        <key type="key" x="69" y="5" characters=";T;Y"/>
        <key type="key" x="101" y="5" characters=";U;I"/>
        <key type="key" x="133" y="5" characters=";O;P"/>

        <key type="key" x="5" y="37" characters=";A;S"/>
        <key type="key" x="37" y="37" characters=";D;F"/>
        <key type="key" x="69" y="37" characters=";G;H"/>
        <key type="key" x="101" y="37" characters=";J;K"/>
        <key type="key" x="133" y="37" characters=";L;"/>

        <key type="key" x="5" y="69" characters=";Z;X"/>
        <key type="key" x="37" y="69" characters=";C;V"/>
        <key type="key" x="69" y="69" characters="; ; "/>
        <key type="key" x="101" y="69" characters=";B;N"/>
        <key type="key" x="133" y="69" characters=";M;"/>
    </keyboardmodel>
```

SpreadKey

```
    <?xml version="1.0" encoding="ISO-8859-1" ?>
    <keyboardmodel name="azerty_spread" width="350" height="136">
```

```
            <variable name="bandwidth" min="1" max="7" step="1" default="5"/>
            <variable name="mainpercentage" min="1" max="100" step="1" default="5"/>
            <variable name="recyclepercentage" min="0" max="100" step="0.1" default="0.1"/>

            <variable name="remapped" default="0"/>
            <variable name="privileged" default="0"/>
            <variable name="keysize" default="0"/>
            <variable name="overriden" default="0"/>
            <variable name="priv_clicked" default="0"/>

            <prediction name="ngram"
                    dictionary="languages/french/ordered-dictionary.txt" />
            <interaction type="dash" event="dash-NO"
                    anglemin="270" anglemax="0" threshold="10"/>
            <interaction type="released" event="released"/>

            <inputmodel name="maincharmodel"
                    fitt_coef="1 / bandwidth"
                    fitt_offset="0"
                    hick_offset="0"
                    hick_items="27"
                    hick_coef="0.2"
                    width=" (H + W) / 2"/>

            <inputmodel name="remappedcharmodel"
                    fitt_coef="1 / bandwidth"
                    fitt_offset="0.3"
                    hick_offset="0"
                    hick_items="27"
                    hick_coef="0.2"
                    width="(H + W) / 2"/>

            <keymodel name="normalkey"

    javaimports="keyboard.prediction.lexicographic.CharacterFrequency;java.awt.*; "
                        width="31" height="31"
                         pressed="SpreadKey.svg#pressed;0;2;30;30;"
                         focused="SpreadKey.svg#idle;3;5;30;30;"
                         over="SpreadKey.svg#over;6;8;30;30;"
                         released="SpreadKey.svg#idle;3;5;30;30;" >
                    <character name="main" svg_suffixe="CENTER" onevent="released" />
                    <character name="secondary" svg_suffixe="NO" onevent="dash-NO" />
                    <function>
    <![CDATA[
            public void contextUpdated () {
                    super.contextUpdated ();
```

```
		EAssistCharacter car = getCharacterByName ("main");
		EAssistCharacter scar = getCharacterByName ("secondary");
		Prediction pred = getPrediction ("ngram");
		if (getProba ("ngram", car.getDefaultValue ()) == 0) {
			double minDist = Double.MAX_VALUE;
			String v = "-";
			for (int i = 0; i < pred.maxP.size (); i++) {
				CharacterFrequency freq = pred.maxP.get (i);
				for (int j = 0; j < keyboard.getKeySet ().size (); j++) {
					EAssistKey k = keyboard.getKeySet ().get (j);
					if (k.getCharacterByName ("main")
							.getDefaultValue ()
							.equals (freq.getLetter ())) {
						double d = distance (k) /
							Math.sqrt (freq.getPercentage ());
						if (d < minDist) {
							minDist = d;
							v = freq.getLetter ();
						}
					}
				}
			}
			car.setCurrentValue (v);
			scar.setCurrentValue (car.getDefaultValue ());
		}
		else {
			car.setDefaultValue ();
			scar.setCurrentValue ("");
		}
	}
	public double getFittsW (String character) {
		if (getCharacterByName ("main").equals (character)) {
			Vector <EAssistKey> neigbours = new Vector <EAssistKey> ();
			neigbours.add (this);
			for (int j = 0; j < neigbours.size (); j++) {
				EAssistKey neig = neigbours.get (j);
				Point c = neig.getCenter ();
				for (int i = 0; i < keyboard.getKeySet ().size (); i++) {
					EAssistKey k = keyboard.getKeySet ().get (i);
					if (k.getCharacterByName ("main")
								.equals (character)) {
						if (k.contains (c.x - w, c.y - h)
						 || k.contains (c.x, c.y - h)
						 || k.contains (c.x + w, c.y - h)
						 || k.contains (c.x - w, c.y)
						 || k.contains (c.x + w, c.y)
```

```
                                                || k.contains (c.x - w, c.y + h)
                                                || k.contains (c.x, c.y + h)
                                                || k.contains (c.x + w, c.y + h)) {
                                                        boolean fnd = false;
                                                        for (int l = 0;
                                                            l < neigbours.size ();
                                                            l ++) {
                                                            if (neigbours.get (l) == k) {
                                                                fnd = true;
                                                                break;
                                                            }
                                                        }
                                                        if (!fnd) {
                                                                neigbours.add (k);
                                                        }
                                                }
                                        }
                                }
                                return super.getFittsW (character) *
                                                Math.sqrt (neigbours.size ());
                        }
                        return super.getFittsW (character);
                }
                public double getFittsB (String character) {
                        if (getCharacter ("main").equals (character)) {
                                return 0.;
                        }
                        else {
                                return 0.2;
                        }
                }
]]>
                        </function>
                </keymodel>

                <key type="normalkey" x="5" y="5" characters="A;A"/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                                y="PREVIOUS_KEY_Y" characters="Z;Z"/>
                <key type="normalkey" x="="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                                y="PREVIOUS_KEY_Y" characters="E;E"/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                                y="PREVIOUS_KEY_Y" characters="R;R"/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                                y="PREVIOUS_KEY_Y" characters="T;T"/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
```

```
                    y="PREVIOUS_KEY_Y" characters="Y;Y"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="U;U"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="I;I"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="O;O"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="P;P"/>
        <key type="normalkey" x="5"
                    y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1" characters="Q;Q"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="S;S"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="D;D"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="F;F"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="G;G"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="H;H"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="J;J"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="K;K"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="L;L"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="M;M"/>
        <key type="normalkey" x="5 + PREVIOUS_KEY_W"
                    y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1" characters="W;W"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="X;X"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="C;C"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="V;V"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                     y="PREVIOUS_KEY_Y" characters="B;B"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                    y="PREVIOUS_KEY_Y" characters="N;N"/>
        <key type="normalkey" x="5 + 2 * PREVIOUS_KEY_W"
                    y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1" characters="; "
                    width="4 * PREVIOUS_KEY_W"/>
    </keybosardmodel>
```

QWERTY Brésilien augmenté

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<keyboardmodel name="azerty_simple" width="360" height="136">

        <interaction type="gesture" event="uppercase" gestures="NORTH"/>
        <interaction type="gesture" event="acute" gestures="SOUTH_WEST"/>
        <interaction type="gesture" event="grave" gestures="SOUTH_EAST"/>
        <interaction type="gesture" event="cedille" gestures="SOUTH"/>

        <interaction type="gesture" event="tild" gestures="N;DOWN_ARROW"/>
        <interaction type="gesture" event="circ" gestures="UP_ARROW"/>
        <interaction type="gesture" event="delete" gestures="WEST"/>
        <interaction type="gesture" event="space" gestures="EAST"/>

        <interaction type="released" event="released"/>

        <keymodel name="normalkey"
                        width="30"
                        height="30">
                <character name="main" svg_suffixe="CENTER"  onevent="released" />
                <character name="uppercase" svg_suffixe=""  onevent="uppercase" />
                <character name="acute" svg_suffixe=""  onevent="acute" />
                <character name="cedille" svg_suffixe=""  onevent="cedille" />
                <character name="grave" svg_suffixe=""  onevent="grave" />
                <character name="circ" svg_suffixe=""  onevent="circ" />
                <character name="tild" svg_suffixe=""  onevent="tild" />
                <character name="delete" svg_suffixe=""  onevent="delete" />
                <character name="space" svg_suffixe=""  onevent="space" />
        </keymodel>

        <key type="normalkey" x="5" y="5" characters="q;(,Q,);;;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                y="PREVIOUS_KEY_Y"
                 characters="w;(,W,);;;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                y="PREVIOUS_KEY_Y"
                characters="e;(,E,);(,é,);;(,è,);(,ê,);;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                y="PREVIOUS_KEY_Y"
                characters="r;(,R,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                y="PREVIOUS_KEY_Y"
                characters="t;(,T,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                y="PREVIOUS_KEY_Y"
```

```
            characters="y;(,Y,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="u;(,U,);(,ú,);;(,ù,);(,û,);;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="i;(,I,);(,í,);;(,ì,);(,î,);;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="o;(,O,);(,ó,);;(,ò,);(,ô,);(,õ,);(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="p;(,P,);;;;;(,,#BACKSPACE); "/>

        <key type="normalkey" x="21" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
            characters="a;(,A,);(,á,);;(,à,);(,â,);(,ã,);(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="s;(,S,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
             y="PREVIOUS_KEY_Y"
            characters="d;(,D,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="f;(,F,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="g;(,G,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="h;(,H,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="j;(,J,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="k;(,K,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="l;(,L,);;;;;(,,#BACKSPACE); "/>

        <key type="normalkey" x="37" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
            characters="z;(,Z,);;;;;(,,#BACKSPACE); "/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y"
            characters="x;(,X,);;;;;(,,#BACKSPACE); "/>
```

```
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                        y="PREVIOUS_KEY_Y"
                        characters="c;(,C,);;(,ç,);;;(,,#BACKSPACE); "/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                        y="PREVIOUS_KEY_Y"
                        characters="v;(,V,);;;;;(,,#BACKSPACE); "/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                        y="PREVIOUS_KEY_Y"
                        characters="b;(,B,);;;;;(,,#BACKSPACE); "/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                        y="PREVIOUS_KEY_Y"
                        characters="n;(,N,);;;;;(,,#BACKSPACE); "/>
                <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                        y="PREVIOUS_KEY_Y"
                        characters="m;(,M,);;;;;(,,#BACKSPACE); "/>

                <key type="normalkey" x="85" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
                        width="3 * PREVIOUS_KEY_W" characters=" ;;;;;;(,,#BACKSPACE); "/>
        </keyboardmodel>
```

Semantickeyboard mobile

```
        <?xml version="1.0" encoding="ISO-8859-1" ?>
        <keyboardmodel name="azerty_simple" width="360" height="136"
                javaimports="keyboard.prediction.lexicographic.CharacterFrequency;java.awt.*;">

                <variable name="size" min="0" max="100" step="1" default="30"/>
                <prediction name="ngram" dictionary="languages/french/dictionary.txt"/>
                <interaction type="released" event="released"/>

                <function>
<![CDATA[
                boolean first = true;
                public void contextUpdated () {
                        super.contextUpdated ();
                        if (first) {
                                Vector keys = getKeySet ();
                                normalkey k = null;
                                for (int j = 0; j < keys.size (); j ++) {
                                        k = (normalkey) keys.elementAt (j);
                                        k.originalPosY = k.getY ();
                                        k.originalPosX = k.getX ();
                                        k.originalPosW = k.getW ();
                                }
                                first = false;
```

```
                }
                else {
                        Vector keys = getKeySet ();
                        for (int j = 0; j < keys.size (); j ++) {
                                sizing ((normalkey) keys.elementAt (j));
                        }
                        Prediction pred = getPrediction ("ngram");
                        for (int i = pred.letterStats.size () - 1; i >= 0; i--) {
                                CharacterFrequency freq =
                                   (CharacterFrequency) pred.letterStats.elementAt (i);
                                String c = freq.getLetter ();
                                for (int j = 0; j < keys.size (); j ++) {
                                        EAssistKey k = (EAssistKey) keys.elementAt (j);
                                        if (k.containCharacter (c)) {
                                                k.raise ();
                                        }
                                }
                        }
                }
                super.contextUpdated ();
        }

        public void sizing (normalkey k) {
                int size = (int) variableSet.get ("size");
                double freq =
                        getFrequency (k.getCharacterByName ("main").getCurrentValue ());
                double freqN = 0;
                double freqS = 0;
                double freqE = 0;
                double freqW = 0;
                double x = k.originalPosX;
                double y = k.originalPosY;
                double w = k.originalPosW;
                double h = size;
                Point p = k.getCenter ();
                EAssistKey N = getKeyXY (p.x, p.y - size);
                EAssistKey S = getKeyXY (p.x, p.y + size);
                EAssistKey E = getKeyXY (p.x + size, p.y);
                EAssistKey W = getKeyXY (p.x - size, p.y);
                if (N != null) freqN =
                        getFrequency (N.getCharacterByName ("main").getCurrentValue ());
                if (S != null) freqS =
                        getFrequency (S.getCharacterByName ("main").getCurrentValue ());
                if (W != null) freqW =
                        getFrequency (W.getCharacterByName ("main").getCurrentValue ());
                if (E != null) freqE =
```

```
                        getFrequency (E.getCharacterByName ("main").getCurrentValue ());
                freqN = 6 * (freq - freqN) / 100;
                freqS = 6 * (freq - freqS) / 100;
                freqE = 6 * (freq - freqE) / 100;
                freqW = 6 * (freq - freqW) / 100;
                y -= freqN;
                h += freqN + freqS;
                x -= freqW;
                w += freqE + freqW;
                k.setSize (x + "", y + "", w + "", h + "");
        }

        public double getFrequency (String ch) {
                Prediction pred = getPrediction ("ngram");
                if (pred.evaluatedSequence.length () == 0) return 0;
                for (int i = pred.letterStats.size () - 1; i >= 0; i--) {
                        CharacterFrequency freq =
                                (CharacterFrequency) pred.letterStats.elementAt (i);
                        String c = freq.getLetter ();
                        if (c.equals (ch)) return freq.getPercentage ();
                }
                return 0;
        }
]]>
        </function>

        <keymodel name="normalkey" width="size" height="size">
                <function>
    <![CDATA[

                public double originalPosX = 0;
                public double originalPosY = 0;
                public double originalPosW = 0;


    ]]>
                </function>
                <character name="main" svg_suffixe="CENTER" onevent="released"/>
        </keymodel>

        <keymodel name="space"
                        width="size"
                        height="size"
                        pressed="SpreadKey.svg#pressed;0;2;30;30;"
                        focused="SpreadKey.svg#idle;3;5;30;30;"
                        over="SpreadKey.svg#idle;6;8;30;30;"
                        released="SpreadKey.svg#idle;3;5;30;30;">
```

```
            <character name="main" svg_suffixe="CENTER" onevent="released"/>
        </keymodel>

        <key type="normalkey" x="5" y="5" characters="Z"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="A"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="R"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="D"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="T"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="U"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="I"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="O"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="P"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="Y"/>

        <key type="normalkey" x="5" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
            characters="Q"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="F"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="S"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="E"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="L"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="G"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="H"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="M"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="J"/>
        <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
            y="PREVIOUS_KEY_Y" characters="K"/>

        <key type="normalkey" x="35" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
            characters="W"/>
```

```
            <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                   y="PREVIOUS_KEY_Y" characters="X"/>
            <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                   y="PREVIOUS_KEY_Y" characters="C"/>
            <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                   y="PREVIOUS_KEY_Y" characters="V"/>
            <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                   y="PREVIOUS_KEY_Y" characters="N"/>
            <key type="normalkey" x="PREVIOUS_KEY_X + PREVIOUS_KEY_W - 1"
                   y="PREVIOUS_KEY_Y" characters="B"/>

            <key type="normalkey" x="65" y="PREVIOUS_KEY_Y + PREVIOUS_KEY_H - 1"
                   width="4 * PREVIOUS_KEY_W" characters=" "/>

        </keyboardmodel>
```

Bibliographie

[Accot 97] Accot, J., Zhai, S . (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '97), Steven Pemberton (Ed.). ACM, New York, NY, USA, 295-302.

[Accot 02] Accot, J., Zhai, S., More than dotting the i's --- foundations for crossing-based interfaces, In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, April 20-25, 2002, Minneapolis, Minnesota, USA.

[Accot 03] J. Accot and S. Zhai. (2003). Refining Fitts' law models for bivariate pointing. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '03). ACM, New York, NY, USA, 193-200.

[Ahlström 05] Ahlström, D.. (2005). Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '05). ACM, New York, NY, USA, 61-70.

[Al Faraj 09a] Al Faraj, K., Mojahid, M., Vigouroux, N., BigKey: A Virtual Keyboard for Mobile Devices, In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part III: Ubiquitous and Intelligent Interaction*, July 19-24, 2009, San Diego, CA.

[Al Faraj 09b] Al Faraj, K., Mojahid, M., Vigouroux, N., 3DKey: An Accordion-Folding Based Virtual Keyboard for Small Screen, In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*, August 24-28, 2009, Uppsala, Sweden.

[Al Faraj 09c] Al Faraj, K., Mojahid, M., Vigouroux, N. SmartKey: A Multi Purpose Target Expansion Based Virtual Keyboard. In *ACM Conference on Computers and Accessibility (ASSETS 2009)*, Pittsburgh, PA, USA,

ACM Press, octobre 2009.

[Appert 08] C. Appert and M. Beaudouin-Lafon. (2008). SwingStates: Adding state machines to Java and the Swing toolkit. *Journal Software Practice and Experience*. 38, 11 (Sep. 2008), 1149-1182.

[Appert 08b] Appert, C., Chapuis, O., and Beaudouin-Lafon, M. (2008). Evaluation of pointing performance on screen edges. In *Proceedings of the working conference on Advanced visual interfaces.* (AVI '08). ACM, New York, NY, USA, 119-126.

[Appert 09] Appert C., Huot S., Dragicevic P. and Beaudouin-Lafon M. Flowstates: Prototypage d'applications interactives avec des flots de données et des machines à états. In *Proceedings of the 21th French speaking conference on Human-Computer Interaction*. (IHM '09), Grenoble, France, October 13 - 16, 2009. ACM Press.

[Arnott 92] Arnott, J.L., Javed, M.Y., (1992). Probabilistic character disambiguation for reduced keyboard using small text samples, *Augmentative and alternative communication, vol. 8*, pp. 215-223.

[Aulagner 10] Aulagner, G., Francois. R., Martin, B., Michel, D., et Raynal, M. (2010). FloodKey: Increasing Software Keyboard Keys by Reducing Needless Ones without Occultation, *Proceedings of 10th WSEAS International Conference on APPLIED COMPUTER SCIENCE*. (ACS '10), Iwate (Japon).

[Badr 09] Badr, G., et Raynal, M. Optimized interaction with word prediction list: A use case with a motor impairment, In *Proceedings of European Conference for the Advancement of Assistive Technology in Europe.* (AAATE 2009), IOS Press, poster, p. 871, Florence (Italie), 31 aout 2009 - 2 septembre 2009.

[Bastien 93] Bastien, J.M.C., Scapin, D. (1993) Ergonomic Criteria for the Evaluation of Human-Computer interfaces. *Institut National de recherche en informatique et en automatique*, France.

[Bellman 98] Bellman, T., and MacKenzie, I. S. A probabilistic character layout strategy for mobile text entry, In *Proceedings of Graphics Interface '98*. Toronto: Canadian Information Processing Society, 1998, 168- 176.

[Benhacène 05] Benhacène, R. (2005). As Rapid as PaperStrips? Evaluation of VertiDigi, a new control tool for Terminal Sectors. In *Proceedings of ATM2005*, Baltimore, USA.

[Benhacène 07] Benhacène R., Merlin B., Rouselle MP. & all, Tackling the problem of flight integration. In *Proceedings of ATM 2007*, Barcelone, Espagne.

[Bi 10] Bi, X., Smith, B. A., and Zhai, S. (2010). Quasi-qwerty soft keyboard optimization. In *Proceedings of the 28th international Conference on Human Factors in Computing Systems*. (Atlanta, Georgia, USA, April 10 - 15, 2010). CHI '10. ACM, New York, NY, 283-286.

[Bier 93] Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and interactive Techniques*. (Anaheim, CA, August 02 - 06, 1993). SIGGRAPH '93. ACM, New York, NY, 73-80.

[Blanch 04] Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. (2004). Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '04). ACM, New York, NY, USA, 519-526.

[Boissière 96] Boissiere, P. and Dours, D. (1996). VITIPI: versatile interpretation of text input by persons with impairments. In *Proceedings of the 5th International conference on Computers helping people with special needs. Part I* (ICCHP '96), Joachim Klaus, Eduard Auff, Willibald Kremser, and Wolfga4.65ng L. Zagler (Eds.). R. Oldenbourg Verlag GmbH, Munich, Germany, Germany, 165-172.

[Boissière 00] Boissière, P. VITIPI : Un système d'aide à l'écriture basé sur un principe d'autoapprentissage et adapté à tous les handicaps moteurs, In In *Proceedings of Handicap 2000*, pp. 81-86, Paris, 2000.

[Buisson 02] Buisson, M. & al. Ivy: Un bus logiciel au service du développement de prototypes de systèmes interactifs, In *Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)* (IHM '02), Michel Beaudouin-Lafon (Ed.). ACM, New York, NY, USA, 223-226.

[Card 78] Card, S. K., English, W. K., and Burr, B. J. 1978. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT. In *Human-Computer interaction: A Multidisciplinary Approach*, R. M. Baecker, Ed. Morgan Kaufmann Publishers, San Francisco, CA, 386-392.

[Card 80] Card, S.K., Moran, T.P., Newell, A. The keystroke-level model for user performance time with interactive systems, *Communications of the ACM*, v.23 n.7, p.396-410, July 1980.

[Card 83] Card, S.K., Newell, A., Moran, T.P. The Psychology of Human-Computer Interaction, *Lawrence Erlbaum Associates, Inc.*, Mahwah, NJ, 1983.

[Castellucci 09] Castellucci, S. J., & MacKenzie, I. S. (2009). TnToolkit: A design and analysis tool for ambiguous, QWERTY, and on-screen keypads. In

Proceedings of the ACM Symposium on Engineering Interactive Computing Systems – EICS 2009, pp. 55-60 New York: ACM.

[Chatty 04] Chatty, S., Sire, S., Vinot, J.L., Lecoanet, P., Lemort, A., Mertz, C., Revisiting visual interface programming: creating GUI tools for designers and programmers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology.* UIST 2004, October 24-27, 2004, Santa Fe, NM, USA.

[Christiernin 05] Christiernin, L.G. Multi-Layered Design - Theoretical Framework and the Method in Practise, in *Winter Meeting 2005 Proceedings,* Department of Computing Science, Chalmers University of Technology, 2005.

[Clark 05] Clark, B., Matthews, J., Deciding Layers: Adaptive Composition of Layers in a Multi-Layer User Interface, In *Proceedings of 11th International Conference on Human-Computer Interaction*, *Volume 7*, July 2005.

[Cockburn 07] Cockburn, A., Gutwin, C., Greenberg, S. (2007). A predictive model of menu performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '07). ACM, New York, NY, USA, 627-636.

[Costagliola 09] Costagliola, G., Di Capua, M., Fuccella, V., Guardi, G. Performances of Multiple-Selection Enabled Menus in Soft Keyboards *DMS'09 - The 15th International Conference on Distributed Multimedia Systems*. Pp. 359-364.

[Darragh 90] Darragh, J. J., Witten, I. H., and James, M. L. 1990. The Reactive Keyboard: A Predictive Typing Aid. *Computer 23*, 11 (Nov. 1990), 41-49.

[Das 08] Das, A. and Stuerzlinger, W. (2008). Modeling learning effects in mobile texting. In *Proceedings of the 7th international Conference on* Mobile and Ubiquitous Multimedia (Umeå, Sweden, December 03 - 05, 2008). MUM '08. ACM, New York, NY, 154-161& Info. Tech., Vol. 20, No. 3, 159-166(8).

[Dragicevic 02] Dragicevic P. and Fekete J-D. ICON: Input Device Selection and Interaction Configuration. Companion *proceedings of the 15th ACM symposium on User Interface Software & Technology*, Paris, France. 27-30 October 2002. pp 47-48.

[Dragicevic 04] Dragicevic P. and Fekete J-D. Support for Input Adaptability in the ICON Toolkit. *Proceedings of the Sixth International Conference on Multimodal Interfaces* (ICMI '04), State College, PA, USA, October 13 - 15, 2004. ACM Press, New York, NY, pp 212-219.

[Dragicevic 05] Dragicevic, P., Chatty, S., Thevenin, D., Vinot, J.L., Artistic resizing: a technique for rich scale-sensitive vector graphics, *Proceedings of the 18th annual ACM symposium on User interface software and technology*, October 23-26, 2005, Seattle, WA, USA.

[Douglas 97] Douglas, S. A. and Mithal, A. K. (1997). The Ergonomics of Computer Pointing Devices. *Springer-Verlag New York, Inc.*

[Dunlop 99] Dunlop, M. D., Crossan, A. Dictionary based text entry method for mobile phones, *Second Workshop on Human-Computer Interaction with Mobile Devices*, Edinburgh, Scotland (1999), 5-7.

[Dvorak 36] Dvorak, A., Merrick, N.L., Dealey, W.L., Ford, G.C.. Typewriting Behavior. *American Book Co.*, New York, USA, 1936.

[Evreinova 04] Evreinova, T., Evreino, G., and Raisamo, R., Four-Key Text Entry for Physically Challenged People, in *the proceedings of 8th ERCIM workshop*, User Interfaces 4 All, Adjunct workshop proceedings, 2004.

[Felzer 05] Felzer, T., Fischer, R., Grönsfelder, T., and Nordmann, R. Alternative control system for operating a PC using intentional muscle contractions only. *In Online-Proc.* CSUN Conf. 2005.

[Felzer 06] Felzer, T., Nordmann, R. Alternative Text Entry Using Different Input Methods, Darmstadt University of Technology, Petersenstr, Germany, in *proceedings of ASSETS'06*, October pp. 22–25, 2006, Portland, Oregon, USA.

[Findlater 10] Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., and Wobbrock, J.O. (2010). Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 153-162.

[Fitts 54] Fitts, P.M., (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381-391.

[Fitts 64] Fitts, P.M, and Peterson, J.R, (1964). Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2):103–112, February 1964.

[Foulds 87] Foulds, R.A., Soede, M., Van Balkom, H., (1987). Statistical disambiguation of multi-character keys applied to reduce motor requirements for augmentative and alternative communication, *Augmentative and alternative communication, vol. 3*, pp. 192-195.

[Furnas 86] Furnas, G.W. 1986. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '86),

Marilyn Mantei and Peter Orbeton (Eds.). ACM, New York, NY, USA, 16-23.

[Gentner 83] Gentner, D. R., Grudin, J. T., Larochelle, S., Norman, D. A., & Rumelhart, D. E. (1983). A glossary of terms including classification of typing errors. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting* (pp. 39-43). New York: Springer-Verlag.

[Gillan 90] Gillan, D. J., Holden, K., Adam, S., Rudisill, M., and Magee, L. 1990. How does Fitts' law fit pointing and dragging?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People* (Seattle, Washington, United States, April 01 - 05, 1990). J. C. Chew and J. Whiteside, Eds. CHI '90. ACM, New York, NY, 227-234.

[Go 07] Go, K. and Endo Y. CATKey: CATKey: customizable and adaptable touchscreen keyboard with bubble cursor-like visual feedback. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction* (INTERACT'07), Cécilia Baranauskas, Philippe Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer-Verlag, Berlin, Heidelberg, 493-496.

[Godard 09] N. GODARD, M. RAYNAL, B. MARTIN, J.-L. VINOT. Etude de l'impact d'une prévisualisation des résultats d'un système de prédiction de caractères. *21th French-speaking conference on Human-computer interaction* (IHM 2009), Grenoble, France, ACM Press, 235-238, 2009.

[Goldberg 93] Goldberg, D., Richardson, C., (1993). Touch-typing with a stylus. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*., pages 168.176. ACM, New York.

[Goldberg 97] Goldberg, D., Unistrokes For Computerized Interpretation of Handwriting. United States Patent 5596656, 1997.

[Gong 08] Gong, J., Tarasewich, P., and MacKenzie, I. S. (2008). Improved word list ordering for text entry on ambiguous keyboards. *Proceedings of the Fifth Nordic Conference on Human-Computer Interaction* - NordiCHI 2008, pp. 152-161. New York: ACM.

[Goodman 02] Goodman, J., Venolia, G., Steury, K., and Parker, C. 2002. Language modeling for soft keyboards. In *Eighteenth national conference on Artificial intelligence*, Rina Dechter, Michael Kearns, and Rich Sutton (Eds.). American Association for Artificial Intelligence, Menlo Park, CA, USA, 419-424.

[Gray 93] Gray, W. D., John, B. E., & Atwood, M. E. Project Ernestine: A validation of GOMS for prediction and explanation of real-world task performance. *Human-Computer Interaction* (1993), 8, 3, 237-309.

[Grinter 01] Grinter, R. E., Eldridge, M., (2001). y do tngrs luv 2 txt msg? In *Proceedings of the 7th European Conference on Computer-Supported Cooperative Work* (ECSCW). Bonn, Germany (Sept. 16-20). 219--238.

[Grinter 03] Grinter, R. E., Eldridge, M., (2003). Wan2tlk?: everyday text messaging, *Proceedings of the SIGCHI conference on Human factors in computing systems*, April 05-10, Ft. Lauderdale, Florida, USA.

[Guiard 09] Guiard, Y., 2009. The problem of consistency in the design of Fitts' law experiments: consider either target distance and width or movement form and scale. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 1809-1818.

[HA 10] HA Hoai Linh, Hody Emmanuelle, Sauvage Mathieu (Plateforme de Conception et d'Evaluation de Système de saisie. Rapport de fin d'étude de Master 2 IHM de Toulouse. 2010.

[Harbusch 03] Harbusch, K, Hasan S., Hoffmann, H., Kuhn, M., Shuler, B. Domain specific Disambiguation for Typing with Ambiguous Keyboards. In *EACL 2003 Workshop on Language Modeling for Text Entry Methods*, Budapest, April 14.

[Hick 52] Hick, W. E. (1952). On the rate of gain of information, *Quarterly Journal of Experimental Psychology*, 4, 11-26.

[Holleis 07] Holleis, P., Otto, F., Hussmann, H., and Schmidt, A. Keystroke-level model for advanced mobile phone interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 1505-1514.

[Hopkins 91] Hopkins, D. (1991). The design and implementation of pie menus. *Dr. Dobb's J.* 16, 12 (Dec. 1991), 16-26.

[Hourcade 08] Hourcade, J.P., Perry, K.B., and Sharma, A. 2008. PointAssist: helping four year olds point with ease. In Proceedings of the 7th *international conference on Interaction design and children* (IDC '08). ACM, New York, NY, USA, 202-209.

[Hourcade 10] Hourcade, J.P., Nguyen, C.M., Perry, K.B., and Denburg, N.L. (2010). Pointassist for older adults: analyzing sub-movement characteristics to aid in pointing tasks. In *Proceedings of the 28th international conference on Human factors in* computing systems (CHI '10). ACM, New York, NY, USA, 1115-1124.

[Huot 04] Huot, S., Dumas, C., et Hégron, G. Svalabard: une table à dessin virtuelle pour la modélisation 3D. *16ème conférence francophone sur l'Interaction Homme-Machine* (IHM 2004), pp. 85-92, Namur, Belgique,

Septembre 2004.

[Hyman 53] Hyman, R. 1953, Stimulus information as a determinant of reaction time, *Journal of Experimental Psychology*, 45, 188-196.

[Igarashi 01] Igarashi, T., Hughes, J.F. A suggestive interface for3D drawing. In *14th Symposium on User Interface Software and Technology* (11-14 Novembre, 2001, Orlando), ACM UIST'01, pp. 173—181.

[Ingmarsson 04] Ingmarsson, M., Dinka, D., and Zhai, S. (2004). TNT: a numeric keypad based text input method. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '04). ACM, New York, NY, USA, 639-646.

[Isokoski 00] Isokoski P. Text input methods for eye trackers using off-screen targets, *Proceedings of the 2000 symposium on Eye tracking research & applications*, p.15-21, November 06-08, 2000, Palm Beach Gardens, Florida, United States.

[Isokoski 01] Isokoski, P. (2001). Model for unistroke writing time. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '01). ACM, New York, NY, USA, 357-364.

[Isokoski 04] Isokoski, P. (2004). Performance of menu-augmented soft keyboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 423-430.

[Isokoski 04b] Isokoski, P. and Raisamo, R. 2004. Quikwriting as a multi-device text entry method. In *Proceedings of the third Nordic conference on Human-computer interaction* (NordiCHI '04). ACM, New York, NY, USA, 105-108.

[Jacob 93] Jacob, R.J.K. Eye Movement-Based Human-computer Interaction Techniques: Toward Non-Command Interfaces, in *H. R. Hartson and D. Hix, editors, Advances in Human- Computer Interaction*, pages 151 - 190, Ablex Publishing Co.,: Norwood, N.J., 1993.

[Jagacinski 85] Jagacinski, R. J., & Monk, D. L. (1985). Fitts' law in two dimensions with hand and head movements. *Journal of Motor Behavior*, 17, 77-95.

[James 01] James, C. L., and Reischel, K. M. Text input for mobile devices: Comparing model predictions to actual performance, *Proceedings of the ACM Conference on Human Factors in Computing Systems* - CHI 2001, ACM, New York, NY (2001), 365-371.

[John 96] John, B. and Kieras, D.E., Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human*

Interaction 3,4 (December 1996a), 287-319.

[Kang 03] Kang, H., Plaisant, C., Shneiderman, B. New approaches to help users get started with visual interfaces: multi-layered interfaces and integrated initial guidance, *Proceedings of the 2003 annual national conference on Digital government research*, pp.1-6, May 18-21, 2003, Boston, MA.

[Koester 94] Koester, H.H. and Levine, S.P. Validation of a keystroke-level model for a text entry system used by people with disabilities. In *Proceedings of the First Annual ACM Conference on Assistive Technologies* (Marina Del Rey, California, United States, October 31 - November 01, 1994). Assets '94. ACM, New York, NY, 115-122.

[Kristensson 07] Kristensson, P.O., Zhai, S., (2007). Learning shape writing by game playing. In *CHI '07 extended abstracts on Human factors in computing systems* (CHI '07). ACM, New York, NY, USA, 1971-1976.

[Kruskal 83a] Kruskal, J.B. (1983). An overview of sequence comparison. In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (D. Sankoff and J.B. Kruskal, eds.) pp. 1-44. Addison-Wesley, Reading, Mass.

[Kruskal 83b] Kruskal, J.B. and Sankoff, D. 1983. n anthology of algorithms and concepts for sequence comparison. In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (D. Sankoff and J.B. Kruskal, eds.) pp. 265-310. Addison-Wesley, Reading, Mass.

[Kurtenbach 91] Kurtenbach, G. and Buxton, W. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 4th Annual ACM Symposium on User interface Software and Technology* (Hilton Head, South Carolina, United States, November 11 - 13, 1991). UIST '91. ACM, New York, NY, 137-144.

[Landauer 85] Landauer, T.K. and Nachbar, D.W. (1985) Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and width. In *Proceedings of ACM CHI 1985 Conference on Human Factors in Computing Systems*, pages 73--78.

[Lesher 98] Lesher, G., Moulton, B., Higginbotham, J. (1998) Optimal Character Arrangements for Ambiguous Keyboards, *IEEE Transactions on Rehabilitation Engineering*, 6, 415-423.

[Lesher 00] Lesher, G.W., and moulton, B.J., (2000). A method for optimizing single-finger keyboards. In *Proceedings of Rehabilitation Engineering Society of North America* - RESNA Annual Conference, pp. 91-93.

[Levenstein 66] Levenstein, V.I., Binary codes capable of correcting deletions,

insertions, and reversals. *Soviet Physics Doklady*, 1966.

[Levine 87] Levine, S.H., Goodenought-Trepagnier, C., Getschow, C.O., Minneman, S.L, (1987). Multi-charater key text entry using computer disambiguation, in *proceedings of the 10th Annual Conference of Rehabilitation Engineering*. pp. 177-179, Washington, DC: RESNA.

[MacKenzie 91] MacKenzie, I. S. (1991). Fitts' law as a performance model in human-computer interaction. Doctoral dissertation (ISBN 0315659858), University of Toronto.

[Mackenzie 92a] Mackenzie, I.S., (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91-139.

[MacKenzie 92b] MacKenzie, I. S., & Buxton, W. (1992). Extending Fitts' law to two-dimensional tasks. *Proceedings of the ACM Conference on Human Factors in Computing Systems* - CHI '92, pp. 219-226. New York: ACM.

[Mackenzie 95] MacKenzie, I. S. (1995). Movement time prediction in human-computer interfaces. In *R. M. Baecker, W. A. S. Buxton, J. Grudin, & S. Greenberg (Eds.), Readings in human-computer interaction (2nd ed.)* (pp. 483-493). Los Altos, CA: Kaufmann.

[MacKenzie 99] MacKenzie, I. S., Zhang, S. X. (1999) The design and evaluation of a high-performance soft keyboard. *Proceedings of the ACM Conference on Human Factors in Computing Systems* - CHI '99, pp. 25-31. New York: ACM.

[MacKenzie 01a] MacKenzie, I.S., Kober, H., Smith, D., Jones, T., Skepner, E., LetterWise: prefix-based disambiguation for mobile text input, *Proceedings of the 14th annual ACM symposium on User interface software and technology*, November 11-14, 2001, Orlando, Florida.

[MacKenzie 01b] MacKenzie, I. S., & Zhang, S. X. (2001). An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology*, 20, 411-418.

[MacKenzie 02a] MacKenzie, I. S. and Soukoreff, R. W., A model of two-thumb text entry, *Proceedings of Graphics Interface 2002*, Toronto: Canadian Information Processing Society (2002), 117--124.

[MacKenzie 02b] MacKenzie, I. S, KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques, *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, p.195-210, September 18-20, 2002.

[MacKenzie 02c] MacKenzie, I. S., & Soukoreff, R. W. (2002). Text entry for mobile computing: Models and methods, theory and practice. *Human-*

Computer Interaction, 17, 147-198.

[MacKenzie 02d] S. MacKenzie. (2002). Mobile text entry using three keys. In *Proceedings of the second Nordic conference on Human-computer interaction* (NordiCHI '02). ACM, New York, NY, USA, 27-34.

[MacKenzie 03] MacKenzie, I. S., & Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems* – CHI 2003, pp. 754-755 New York: ACM.

[MacKenzie 09] MacKenzie, I. S. The one-key challenge: searching for a fast one-key text entry method. In *Proceedings of the 11th international ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, Pennsylvania, USA, October 25 - 28, 2009). Assets '09. ACM, New York, NY, 91-98.

[Magnien 04] Magnien, L., Bouraoui, J.L., and Vigouroux, N. (2004). Mobile devices: soft keyboard text-entry enhanced by Visual Cues. In *Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing* (UbiMob '04). ACM, New York, NY, USA, 158-165.

[Mason 00] Mason, S.G., Birch, G.E. A Brain-Controlled Switch for Asynchronous Control Applications. *IEEE Trans. Biomed. Eng.*, 47(10):1297--1307, 2000.

[Majaranta 02] Majaranta, P., Räihä, K-J 2002. Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications* (ETRA '02). ACM, New York, NY, USA, 15-22.

[Majaranta 07] Majaranta, P., Räihä, K-J. (2007) Text Entry by Gaze: Utilizing Eye-Tracking.In I.S. MacKenzie & K. Tanaka-Ishii (Eds.), *Text entry systems: Mobility, accessibility, universality* (pp. 175-187). San Francisco: Morgan Kaufmann. ISBN: 978-0-12-373591-1.

[Martin 05] Martin, B., (2005) VirHKey: a VIRtual Hyperbolic KEYboard with gesture interaction and visual feedback for mobile devices, *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, Salzburg, Austria.

[Martin 07] B. Martin, I. Pecci. Etat de l'art des claviers physiques et logiciels pour la saisie de texte. *Revue d'Interaction Homme-Machine (RIHM)*, n°2, 147-205, 2007.

[Martin 08] Martin, B., Isokoski, P., Edgewrite with Integrated Corner Sequence Help. *SIGCHI Conference on Human Factors in Computing Systems* (CHI '08), Florence, Italy, April 5-10, ACM Press, 583-592, 2008.

[Martin 09] Martin, B., Isokoski, P., Jayet, F., and Schang, T. 2009. Performance of finger-operated soft keyboard with and without offset zoom on the pressed key. In *Proceedings of the 6th international Conference on Mobile Technology, Application & Systems* (Nice, France, September 02 - 04, 2009). Mobility '09. ACM, New York, NY, 1-8.

[Masui 99] Masui, T. POBox: An Efficient Text Input Method for Handheld and Ubiquitous Computers. In *proceedings of the International Symposium on Handheld and Ubiquitous Computing* (HUC99), September, 1999, pp. 289-300.

[Matias 96] Matias, E., MacKenzie, I.S. and Buxton, W. (1996). One-handed touch typing on a QWERTY keyboard. *Human-Computor Interact.* 11, 1 (March 1996), 1-27.

[McQueen 95] McQueen, C., MacKenzie, I. S., and Zhang, S. X. (1995). An extended study of numeric entry on pen-based computers. *Proceedings of Graphics Interface '95*, pp.215-222. Toronto: Canadian Information Processing Society.

[Merlin 04] Merlin B., Privat R., Raynal M., Truillet P. Enseignement et ingénierie des interfaces multimodales : une expérience toulousaine. In *Proceedings of IHM 2004*. Namur, Belgique.

[Merlin 07] Merlin B., Benhacène R., Kapp, V., (2007). Interface multi-layer et processus d'évolution des systèmes interactifs en activité critique. In *Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine* (IHM '07). ACM, New York, NY, USA, 191-198.

[Merlin 08] Merlin B., Hurter C., Benhacène R., (2008). A solution to interface evolution issues: the multi-layer interface. In *CHI '08 extended abstracts on Human factors in computing systems* (CHI '08). ACM, New York, NY, USA, 2715-2720.

[Merlin 09a] Merlin, B., Raynal, M., SpreadKey: increasing software keyboard key by recycling needless ones. In *proceedings of AAATE 2009*, Florence, Italia.

[Merlin 10a] Merlin, B., Raynal, M., Evaluation of SpreadKey system with motor impaired users. In *Proceedings of the 12th international conference on Computers helping people with special needs* (ICCHP'10), Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer (Eds.). Springer-Verlag, Berlin, Heidelberg, 112-119.

[Merlin 10b] Merlin B., Raynal M., Soft Keyboard evaluations: Integrating user's background in predictive models, in *proceedings of IADIS International Conference Interfaces and Human Computer Interaction 2010*, Freiburg, Germany, 28 - 30 July 2010.

[Merlin 10c] Merlin B., Hirata, C., Experiência de projeto de sistemas colaborativos para atividades críticas: controle de tráfego aéreo. *Simpósio Brasileiro de Sistemas Colaborativos*, October 5-8, 2010, Belo Horizonte, Brazil.

[Miró-Borrás 09] Miró-Borrás, J. and Bernabeu-Soler, P. (2009). Text entry in the e-commerce age: two proposals for the severely handicapped. J. *Theor. Appl. Electron. Commer. Res*. 4, 1 (Apr. 2009), 101-112.

[Nesbat 03] Nesbat, S. B. (2003). A System for Fast, Full-Text Entry for Small Electronic Devices. *Proceedings of the Fifth International Conference on Multimodal Interfaces*, ICMI 2003 (ACM-sponsored), Vancouver.

[Nielsen 90] Nielsen, J., Molich, R., (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people* (CHI '90), Jane Carrasco Chew and John Whiteside (Eds.). ACM, New York, NY, USA, 249-256.

[Nielsen 93] Nielsen, J., Phillips, V.L. (1993). Estimating the relative usability of two interfaces: heuristic, formal, and empirical methods compared. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (CHI '93). ACM, New York, NY, USA, 214-221.

[Nielsen 94] Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence* (CHI '94), Beth Adelson, Susan Dumais, and Judith Olson (Eds.). ACM, New York, NY, USA, 152-158.

[Norman 82] Norman D.A., Fisher, D. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors*, 24:509_515, 1982.

[Pavlovych 03] Pavlovych, A., & Stuerzlinger, W. Less-Tap: A fast and easy-to-learn text input technique for phones, *Proceedings of Graphics Interface* – GI 2003. Canadian Information Processing Society (2003).

[Pavlovych 04] Pavlovych, A. and Stuerzlinger, W. (2004). Model for non-expert text entry speed on 12-button phone keypads. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 351-358.

[Pavlovych 05] A. Pavlovych, W. Stuerzlinger (2005). An Analysis of Novice Text Entry Performance on Large Interactive Wall Surfaces, In *proceedings of HCI International 2005*, Ed. G. Salvendy, Lawrence Erlbaum.

[Perlin 98] Perlin, K., Quikwriting: continuous stylus-based text entry, Proceedings of the 11th annual ACM symposium on User interface software and

technology, p.215-216, November 01-04, 1998, San Francisco, California, United States.

[Poirier 06] Poirier, F., Belatar, M., Glyph 2: une saisie de texte avec deux appuis de touche par caractère - principes et comparaisons, *Proceedings of the 18th international conference on Association Francophone d'Interaction Homme-Machine*, p.159-162, April 18-21, 2006, Montreal, Canada.

[Poirier 07] Poirier, F., and Sad, H.H. (2007). A new performance measure taking into account the mental load in mobile text entry tasks. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II* (INTERACT'07). Springer-Verlag, Berlin, Heidelberg, 653-656.

[Poirier 08] Poirier F., Sad H. *A platform for mobile text entry methods evaluation*. Int. Conf. IRIA ACHI 2008.

[Raghunath 02] Raghunath, M. T. and Narayanaswami, C., User Interfaces for Applications on a Wrist Watch, *Persona7-30l and Ubiquitous Computing*, 6, (2002.), pp. 17-30.

[Raynal 05a] Raynal., M., Vigouroux, N., KeyGlasses: Semi-transparent keys to optimize text input on virtual keyboard, In *Proceedings of AAATE'05*, Lille, 6-9 septembre 2005, p.713-717.

[Raynal 05b] Raynal, M. and Vigouroux, N. (2005). Genetic algorithm to generate optimized soft keyboard. In *CHI '05 extended abstracts on Human factors in computing systems* (CHI '05). ACM, New York, NY, USA, 1729-1732.

[Raynal 05c] Raynal, M., Maubert, S., Vigouroux, N., Vella, F., Magnien, L. E-ASSISTE: A Platform Allowing Evaluation of Text Input Systems, In *proceedings of UAHCI'05*, Las Vegas (U.S.A), 22-27 juillet 2005.

[Raynal 05d] Raynal, M. : Systèmes de saisie de textes pour les personnes handicapées moteur : optimisation, interaction et mesure de l'utilisabilité, thèse de doctorat, IRIT, 2005.

[Raynal 07a] Raynal., M., Vinot, J-L, and Truillet, P., Fisheye Keyboard: Whole Keyboard Displayed on Small Device. *20th ACM UIST Symposium*, 07-10 October 2007, Newport (USA).

[Raynal 07b] Raynal., M., and Truillet, P.,Fisheye Keyboard: Whole Keyboard Displayed on PDA, *HCI International 2007*, Beijing (China), 22-27 July 2007, HCII 2007, Part II, LNCS 4551, pp. 452-459, Springer-Verlag 2007.

[Raynal 07c] Raynal., M., (2007) Le système KeyGlass : Système d'ajout dynamique de touches sur clavier logiciel, *numéro spécial de la revue TAL : la communication assistée*.

[Ravden 89] Ravden S., Johnson, G. Evaluating Usability of Human-Computer Interfaces, a practical method. Ellis Horwood Limited, Chichester, 1989.

[Rochester 78] Rochester, N., Bequaert, F.C., Sharp, E.M. 1978. The Chord Keyboard. *Computer* 11, 12 (December 1978), 57-63.

[Sandnes 04] Sandnes, F. E., Thorkildssen, H. W., Arvei, A., and Buverud, J. O. 2004. Techniques for Fast and Easy Mobile Text-Entry with Three-Keys. In *Proceedings of the 37th Annual Hawaii international Conference on System Sciences* (Hicss'04) - Track 9 - Volume 9 (January 05 - 08, 2004). HICSS. IEEE Computer Society, Washington, DC, 90286.2.

[Sandnes 05] Sandnes, F. E. (2005). Evaluating mobile text entry strategies with finite state automata. In *Proceedings of the 7th international Conference on Human Computer interaction with Mobile Devices & Services* (Salzburg, Austria, September 19 - 22, 2005). MobileHCI '05, vol. 111. ACM, New York, NY, 115-121.

[Schadle 04] Schadle, I., Antoine, J.-Y., Le Pévédic, B., Poirier, F., *Sibyl - AAC system using NLP techniques . 9th International Conference on Computers Helping People with special needs*, ICCHP'2004, LNCS 3118, p. 1009-1015, Paris.

[Sears 01] Sears A., Jacko J. A., Chu J., & Moro F. (2001). The role of visual search in the design of effective soft *keyboards. Behaviour & information technology.*

[Silfverberg 00] Silfverberg, M., MacKenzie, I. S., and Korhonen, P. Predicting text entry speed on mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM, New York, NY, 9-16.

[Shannon 48] Shannon, C., (1948). A Mathematical Theory of Communication. *Bell System Technical Journal 27* (July and October): pp. 379–423, 623–656.

[Shneiderman 03] Shneiderman, B., Promoting universal usability with multi-layer interface design, *Proceedings of the 2003 conference on Universal usability*, pp.1-8, November 10-11, 2003, Vancouver, British Columbia, Canada.

[Soukoreff 95] Soukoreff, R.W., MacKenzie, I.S. Theoretical upper and lower bounds on typing speeds using a stylus and soft keyboard. *Behaviour & Information Technology*, 14 (1995), 370-379.

[Soukoreff 01] Soukoreff, R.W. & MacKenzie, I.S. (2001). Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems* - CHI 2001, 319-320.

[Soukoreff 03] Soukoreff, R.W. & MacKenzie, I.S. (2003). Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric *Proceedings of the ACM Conference on Human-Factors in Computing Systems* - CHI 2003, 113-120.

[Soukoreff 04] Soukoreff, R.W. & MacKenzie, I.S. (2004). Recent developments in text-entry error rate measurement, *Extended Abstracts of the ACM conference on Human Factors in Computing Systems* - CHI 2004, 1425-1428.

[Tanaka-Ishii 02] Tanaka-Ishii, K., Inutsuka, Y., and Takeichi, M. (2002). Entering text with a four-button device. In *Proceedings of the 19th international Conference on Computational Linguistics - Volume 1* (Taipei, Taiwan, August 24 - September 01, 2002). International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, 1-7.

[Tanaka-ishii 07] Tanaka-ishii, K. (2007). Word-based predictive text entry using adaptive language models. *Nat. Lang. Eng.* 13, 1 (March 2007), 51-74.

[Textware 98] Textware Solutions. (1998). http://fitaly.com/fitaly/fitaly.htm

[Trnka 06] Trnka. K., Yarrington, D., McCoy, K., and Pennington, C.. 2006. Topic modeling in fringe word prediction for AAC. In *Proceedings of the 11th international conference on Intelligent user interfaces* (IUI '06). ACM, New York, NY, USA, 276-278.

[Trnka 09] Trnka, K., McCaw, J., Yarrington, D., McCoy, K.F., and Pennington, C. (2009). User Interaction with Word Prediction: The Effects of Prediction Quality. *ACM Trans. Access. Comput.* 1, 3, Article 17 (February 2009), 34 pages.

[US6307549 95] Reduced keyboard disambiguating system, Patent -US6307549, (1995).

[Vella 04] Vella, F., Vigouroux, N., Truillet, P. (2004). Environnement de conception de clavier virtuel: SOKEYTO (SOftware KEYboard TOolkit). In *Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine* (IHM 2004). ACM, New

York, NY, USA, 181-182.

[Vella 05] Vella, F., Collignon, A., David, A., Chabbert, V., Vigouroux, N. (2005). Pour une meilleure utilisabilité des pages web par des handicapés moteurs: modèle de Fitts et méthodes de conception centrée-utilisateur. In *Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine* (IHM 2005). ACM, New York, NY, USA, 239-242.

[Vigouroux 04] Vigouroux, N., Vella, F., Truillet, P. et Raynal, M. Evaluation of AAC for Text Input by Two Groups of Subjects: Able-bodies Subjects and Disabled Motor Subject, In *proceedings of 8th ERCIM UI4All*, Vienne (Autriche), 28-29 juin 2004.

[Wandmacher 07] Wandmacher, T., Antoine, J-Y., Poirier, F., SIBYLLE: A System for Alternative Communication Adapting to the Context and Its User, *proceedings of ASSETS'07*, October 15–17, 2007, Tempe, Arizona, USA, pp. 203-210.

[Wandmacher 08] Wandmacher, T., Antoine, J., Poirier, F., and Départe, J. (2008). Sibylle, An Assistive Communication System Adapting to the Context and Its *User. ACM Trans. Access. Comput. 1*, 1 (May. 2008), 1-30.

[Ward 00] Ward, J. D., Blackwell, A. F., MacKay, D. J. C., (2000). Dasher- a data entry interface using continuous gesture and language models. In *Proceedings of the 13th annual ACM symposium on User Interface Software and Technology*. UIST'00, ACM Press, New York, NY, USA, pp. 129-137.

[Wigdor 04] Wigdor, D., Balakrishnan, R., A comparison of consecutive and concurrent input text entry techniques for mobile phones, *Proceedings of the SIGCHI conference on Human factors in computing systems*, p.81-88, April 24-29, 2004, Vienna, Austria.

[Wobbrock 03] Wobbrock, J. O., Myers, B. A., and Kembel, J. A. (2003). EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User interface Software and Technology* (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM, New York, NY, 61-70.

[Wobbrock 06a] Wobbrock, J. O., Myers, B. A 2006. From letters to words: efficient stroke-based word completion for trackball text entry. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility* (Assets '06). ACM, New York, NY, USA, 2-9.

[Wobbrock 06b] Wobbrock, J. O. and Myers, B. A. 2006a. Analyzing the Input Stream for Character-Level Errors in Unconstrained Text Entry Evaluations. Transactions on Computer-Human Interaction 13, 4, 458–489.

[Woodworth 99] Woodworth, R.S, The accuracy of voluntary movement. *Psychological*

review, 3(2): 1-4, 1899.

[Yang 10] Yang, H. and Xu, X. (2010). Bias towards regular configuration in 2D pointing. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, New York, NY, USA, 1391-1400.

[Zhai 00] Zhai, S. , Hunter, M., Smith, B.A., The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design, *Proceedings of the 13th annual ACM symposium on User interface software and technology*, p.119-128, November 06-08, 2000, San Diego, California, United States.

[Zhai 01] Zhai, S. and Smith, B.A. (2001). Alphabetically biased virtual keyboards are easier to use: layout does matter. In *CHI '01 extended abstracts on Human factors in computing systems* (CHI '01). ACM, New York, NY, USA, 321-322.

[Zhai 02a] Zhai, S., Hunter, M., Smith, B.A., Performance Optimization of Virtual Keyboards, *Human- Computer Interaction. Vol. 17*, No2&3. 2002. pp 229-269.

[Zhai 02b] Zhai, S., Sue, A., and Accot, J. (2002). Movement model, hits distribution and learning in virtual keyboarding. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (CHI '02). ACM, New York, NY, USA, 17-24.

[Zhai 03] Zhai, S., Kristensson, P.O. **(**2003). Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '03). ACM, New York, NY, USA, 97-104.

[Zhai 08] Zhai, S. and Kristensson, P. O., Interlaced QWERTY - accommodating ease of visual search and input flexibility in shape writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (CHI'08), ACM, New York, NY, 593-596.

[Zhai 09] Zhai, S., Kristensson, P., Gong, P., Greiner, M., Peng, S. A., Liu, L. M., and Dunnigan, A. 2009. Shapewriter on the iphone: from the laboratory to the real world. In *Proceedings of the 27th international Conference Extended Abstracts on Human Factors in Computing Systems* (Boston, MA, USA, April 04 - 09, 2009). CHI '09. ACM, New York, NY, 2667-2670.

[Zhang 98] Zhang, S.X., A high performance soft keyboard for mobile systems. Master's thesis, The University of Guelph, 1998.

Résumé

Avec l'expansion des dispositifs mobiles, l'efficacité de la saisie de texte est un défi de plus en plus important pour l'interaction homme-machine. Or, nous observons que, bien que les claviers type AZERTY ou téléphone, traditionnellement utilisés sur ces supports, soient évalués comme sous-optimaux, et, bien que de nombreuses alternatives évaluées comme plus performantes soient proposées dans la littérature, ces nouvelles alternatives restent très marginalement utilisées. Sur la base de cette observation, nous argumentons que la finalité des évaluations ne tient compte que d'un aspect du clavier, aspect qui n'est pas représentatif de la capacité d'un utilisateur à intégrer les concepts proposés dans son quotidien. Nous proposons en conséquence une stratégie complémentaire d'évaluation sur la base d'une évaluation heuristique des claviers logiciels.

Par ailleurs, de manière à faciliter la mise en œuvre des évaluations et simplifier le design de nouveaux claviers, nous proposons une nouvelle version (E-Assist II) de la plate-forme E-Assiste. Elle permet, en premier lieu, de facilité le design et le déroulement des expérimentations, et plus largement d'encadrer les évaluations théoriques, expérimentales et heuristiques des claviers. Une version TinyEAssist permet également de déployer des expérimentations sur des supports mobiles (téléphones portables notamment). En second lieu, sur la base de l'étude de la structure des claviers logiciels, nous avons de plus proposé un langage de spécification des claviers permettant de générer des claviers logiciels complexes (interagissant potentiellement avec des systèmes de prédiction) à des fins d'expérimentation ou de simple usage.

Enfin, sur la base des critères de performance mis en évidence par les évaluations heuristiques, nous proposons quatre nouveaux paradigmes de claviers. Parmi ces paradigmes deux ont offert des perspectives particulièrement intéressantes : en premier lieu le clavier multi-layer consistant à conduire progressivement, au cours d'une période transitoire, un utilisateur d'une distribution de touches type AZERTY vers une distribution de touches optimisée ; Le second consistant à faciliter l'accès aux caractères type accents, majuscules ou ponctuation, souvent déconsidérés dans l'optimisation des claviers logiciels.

Abstract

The expansion of mobile devices turn text input performances a major challenge for Human-Machine Interaction. We observed that, even if traditional QWERTY soft keyboards or telephone based soft keyboard were evaluated as poorly efficient, and, even if several alternatives evaluated as more efficient were proposed in the research field, these new alternatives are rarely used. Based on this observation, we argue that the goal of soft keyboard evaluation focus on long term performances whereas does not take into account the perspective for a user to use it in his quotidian. Consequently, we propose a complementary evaluation strategy base on heuristic evaluation methodology.

In order to ease the evaluation and design of new soft keyboards, we proposed a new version (E-Assist II) of the E-Assiste plate-form. This plate-form aims, at first, to facilitate the design and procedure of experimentations and, more generally, to guide the theoretical, experimental and heuristic evaluations. A compact version (TinyEAssist) enables to perform experimentation on mobile environment such as mobile phone. At second, based on soft keyboard structure study, we proposed a keyboard specification language enabling to generate complex keyboard (including soft keyboard interacting with prediction systems). The generated soft keyboards may be used into the experimentation plate-form or interacting with the exploration system.

At last, based on the criteria highlighted by the heuristic evaluation, we proposed four new soft keyboard paradigms. Among them two paradigms showed interesting perspectives: at first the multilayer keyboard consist in accompanying the user from a standard QWERTY layout to an optimized layout during a transition period; the second consist in accelerating the access to the characters such as accents, upper-case, punctuation, etc., frequently ignored in the keyboard optimizations.