MenuDfA : navigation gestuelle tactile sans contrainte dans un menu linéaire hiérarchique

Eric Petit, Denis Chêne

MenuDfA : navigation gestuelle tactile sans contrainte dans un menu linéaire hiérarchique

Eric Petit
Orange
38240, Meylan, France
eric.petit@orange.com

Denis Chêne
Orange
38240, Meylan, France
denis.chene@orange.com

RESUME
Cet article présente une nouvelle technique de menus linéaires appelée MenuDfA dont l'originalité réside dans une navigation gestuelle qui relâche la contrainte visio-motrice de positionnement absolu. Cette technique est adaptée à la sélection de commandes sur dispositifs mobiles à écran tactile, notamment lorsque l'attention visuelle de l'utilisateur est partagée. Elle s'applique aussi à des listes d'items quelconques et de longueur variable. Dans cette approche, l'utilisateur parcourt l'arborescence du menu en manipulant un focus de sélection à l'aide de gestes simples pouvant démarrer n'importe où sur l'écran. On introduit un style de gestes basé sur du positionnement relatif couplé à des retours visuels innovants. Cette gestuelle reste compatible avec le pointage absolu et permet en outre la composition spatiale, rendant la navigation fluide.

Mots Clés
Technique de menus ; interaction gestuelle ; smartphone.

ACM Classification Keywords
H.5.2 User Interfaces: *Interaction styles, Prototyping,* I.3.6 *Interaction techniques*

INTRODUCTION

Naviguer au doigt dans une interface tactile est devenu une tâche courante depuis l'avènement des téléphones mobiles à écran capacitif (ou smartphones), tels que l'iPhone d'Apple et les terminaux Android de Google. Ce type d'interface reprend les fondements des interfaces graphiques interactives (Xerox Star). Le style d'interaction, basé sur la manipulation directe est décrit par Shneiderman [17]. Ce style repose notamment sur une représentation permanente des objets d'intérêt, l'utilisation d'actions physiques (e.g. pointage et sélection à la souris) plutôt que de commandes à la syntaxe complexe, des opérations rapides dont les effets doivent être immédiatement visibles sur les objets. Sur dispositifs mobiles, cette interaction tactile se limite bien souvent à du pointage (par exemple de type appui bref) ou à une gestuelle simple (de type mouvement rectiligne horizontal ou vertical) en couplage fort avec les objets affichés. Ces derniers, organisés spatialement, reprennent l'apparence d'objets réels (boutons virtuels, page-écran, objets, etc.) incitant l'utilisateur à agir selon des procédures qui lui font sens [12] : un onglet va ainsi passer au premier plan, une page se tourner ou se dérouler en fonction de son apparence. La richesse graphique de ces interfaces, conséquence des progrès technologiques en termes matériel et logiciel, oriente l'interaction utilisateur sur le pointage et sur une manipulation tactile fortement dépendante du repérage visuel. Ce style de manipulation ne prend pas vraiment en compte les multiples situations d'usage. En effet, les utilisateurs peuvent être en situation de contrainte visuelle (personne mal ou non-voyante, ou personne standard en plein soleil), motrice (personne à manipulation réduite, ou personne standard utilisant son téléphone tout en marchant) et/ou cognitive (conducteur qui a les yeux, les pieds et les mains bien occupés et qui, tout en conduisant, règle néanmoins sa radio). A cela s'ajoute les difficultés d'usage inhérentes à l'utilisation de dispositifs mobiles à écran tactile de petite taille [19, 16]. Les personnes qui sont en situation de ne pas pouvoir contrôler visuellement leur interaction et de manipuler finement l'interface doivent donc changer de logique d'interaction. Pour toutes ces situations d'usage il est nécessaire de proposer autre chose que de la manipulation directe à fort contrôle visuo-moteur.

Nous décrirons dans cet article, dans la continuité de l'approche «interfaces utilisateurs pour tous» de Stephanidis [18], des principes universels de navigation hiérarchique permettant de concevoir des interfaces utilisables dans les différents contextes, de dispositifs, d'usages, et d'individus. Nous proposerons ainsi des objets d'interfaces qui soient utilisables dans deux logiques d'interaction : la manipulation de l'objet avec contrôle visuel (navigation par pointage), et la manipulation de l'objet sans contrôle visuel (navigation par positionnement relatif). Plus précisément, nous présentons ici une nouvelle technique de navigation au doigt dans un menu linéaire hiérarchique, nommée MenuDfA (DfA pour Design for All). Cette technique de menus, basée sur le geste, permet une utilisation efficace des représentations arborescentes dans le cas d'une interaction sur petit écran de type smartphone. Elle est de plus adaptée à une situation à faible contrôle visuo-moteur. Ces travaux sont issus de deux axes de recherche. Le premier porte sur la reconnaissance de

NB : Cet article fait la synthèse de plusieurs années de recherche menées en interne chez Orange Labs. Il a été soumis à la conférence IHM 2014. Il a été rejeté aux motifs qu'il était peu standard, qu'il manquait de validations expérimentales, et parce que les relecteurs ont trouvé que notre technique était trop similaire à celle des Marking Menus, ce que nous réfutons.

gestes et la mise au point d'un *framework* d'interaction gestuelle tactile nommé DGIL[1]. Le deuxième, porte sur la prise en compte du handicap visuel et moteur. Il a guidé nos choix de conception ergonomique, en particulier le choix des gestes et des règles de navigation. A souligner que notre technique est applicable aux menus tels que définis dans [2, 11], mais aussi, plus largement, aux listes à nombre d'items variable, défilantes ou non, rémanentes ou non (par exemple un menu d'options). En conséquence, elle n'impose pas que tous les éléments de la liste courante soient visibles et donc atteignables par pointage direct. Ce faisant, elle relâche la contrainte sur la largeur de l'arborescence. Selon l'analyse MenUA [1] le nombre maximum de commandes offertes par le menu est un *facteur d'applicabilité* du menu, c'est donc une caractéristique importante des menus. Notre approche dépasse donc le strict problème de la sélection de commandes. Elle vise à fournir un composant générique de navigation gestuelle susceptible de servir de composant central d'une application.

La section ci-après, décrit les principes et sources d'inspiration qui sont à la base de cette technique. Elle est suivie par une section portant sur la réalisation du menu MenuDfA en tant que composant logiciel d'interface sur plateforme mobile Android. Des prototypes d'application seront mentionnés ainsi que des résultats de test utilisateur.

PRINCIPES

De l'arbre logique des commandes aux menus linéaires hiérarchiques

Interagir, c'est effectuer une succession de choix, au sein de listes d'items, de commandes. Pour ce faire l'utilisateur doit sélectionner et déclencher ces items. Ces listes peuvent être présentées de façon horizontale ou verticale. Lors d'une activité de lecture ces dernières sont plus efficaces et sont donc à privilégier [7]. Par ailleurs, généralement les objets d'interfaces sont très nombreux et il est nécessaire de naviguer au sein de plusieurs listes, imbriquées les unes dans les autres. Elles sont caractérisées par un arbre logique à l'arborescence (voir Figure 1) plus ou moins profonde. Ce qu'il nous faut définir de façon universelle c'est la navigation au sein de la représentation de cette arborescence. Ce type de hiérarchie de listes se retrouve dans les menus cascadés, typiquement la barre de menu sous MAC OS d'Apple. Une version de cette technique, adaptée à la manipulation au doigt et optimisant l'espace d'affichage, a été proposée par Kobayashi [9]. Cette dernière, en prenant en compte la direction du mouvement du pointeur réduit les erreurs de manipulation liées à la contrainte du « couloir horizontal de déplacement ». Ce faisant, elle introduit le geste dans l'interaction.

[1] Dynamic Gesture Interaction Layer [13]

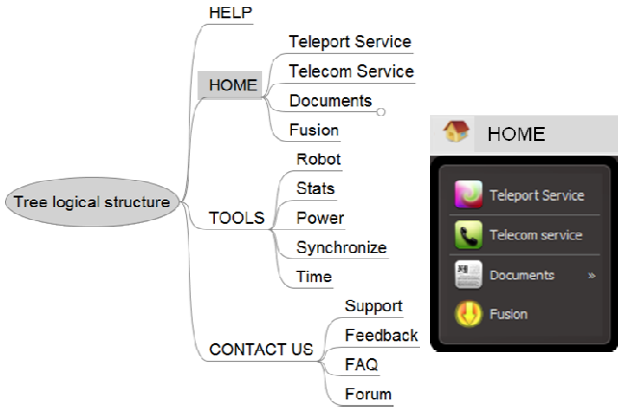

Figure 1: représentation logique des fonctions et représentation visuelle au sein d'un menu linéaire hiérarchique

Notre approche en reprend certains principes et conventions gestuelles, comme l'utilisation des deux dimensions : orientation et distance, ainsi que les quatre commandes : des gestes verticaux pour déplacer la sélection locale vers le bas ou vers le haut, un geste vers la droite pour ouvrir le sous-menu (ou valider/invalider une option), un geste vers la gauche pour fermer le sous-menu et revenir au menu parent. Ces mêmes quatre opérations se retrouvent dans les listes hiérarchiques zoomables de Lecolinet et al. [8] mais avec une gestuelle plus fine où le mouvement du pointeur est interprété en continu, du-moins selon l'axe horizontal, ceci en tirant partie de la technique du Control Menu [15]. En particulier, lorsque l'utilisateur effectue un mouvement horizontal vers la droite sur un item de son choix (en tant que sous-menu), il fait apparaître graduellement son contenu (les nœuds fils) jusqu'à ce que la sous-liste s'affiche pleinement. Le geste opposé vers la gauche réalise l'opération inverse en repliant les branches du nœud. La technique MenuDfA réutilise ces principes, avec toutefois trois différences notables : (1) elle met en œuvre une interaction gestuelle indépendante de la représentation graphique (ou déspatialisée) car fondée sur un parcours logique de navigation, (2) elle permet l'utilisation conjointe ou séparée des deux types de positionnement, absolu et relatif, (3) et seul le menu courant nécessite d'être affiché. La logique séquentielle est basée sur un focus de sélection que l'utilisateur manipule avec des gestes simples ou composés. Cette logique est empruntée aux interfaces dites accessibles, comme discuté ci-après.

Logique de navigation séquentielle

Avant la souris, le clavier permettait déjà de sélectionner les objets d'interface. Mais la prédominance du pointage a fortement réduit l'usage de la logique d'accès séquentiel. Cette dernière est cependant fortement soutenue dans le cadre des normes d'accessibilité du Web (W3C-WAI). Ainsi, à l'aide de la touche « Tab » du clavier d'ordinateur, l'utilisateur peut parcourir sa page Web uniquement en déplaçant un focus sur les différents éléments de la page. Les éléments reçoivent

alors le focus dans un ordre séquentiel respectant la logique du contenu. Les personnes ayant des limitations motrices (ne leur permettant pas d'utiliser la souris) ou ayant des limitations visuelles sévères utilisent ce mode séquentiel. Dans cette approche, un des éléments du parcours possède le focus qui reste visible de façon permanente, ce qui est équivalent à une sélection permanente. Dès lors, l'activation (ou la validation) de la sélection courante peut avoir lieu de manière séparée, spatialement et temporellement. Cette caractéristique fait que cette manière de naviguer dans les interfaces est également très utilisée dans les dispositifs qui utilisent une interaction indirecte, comme par exemple dans les interfaces TV contrôlées par une télécommande. Mais concernant les interfaces tactiles modernes, seul le mode *accessible* exploite ce style de navigation, avec néanmoins certaines limitations gestuelles liées au mode d'exploration spatial, comme expliqué ci-après.

Interfaces tactiles conçues pour tous

Rappelons tout d'abord une caractéristique des interfaces tactiles modernes de type smartphone ou tablette. Leur modèle d'interaction, hérité du modèle WIMP (en anglais « Windows, Icones, Menus, Pointer »), repose principalement sur le pointage direct des objets, étant entendu que chaque élément d'interface possède une position spatiale précise et stable que l'utilisateur doit repérer visuellement s'il veut interagir avec. Il y a alors coïncidence entre l'espace moteur et l'espace visuel. Ce modèle d'interaction, basé sur du positionnement absolu, est également exploité dans les interfaces tactiles dédiées aux personnes non voyantes, selon un mode d'interaction dit « exploratoire ». Dans ce mode, la personne non voyante explore spatialement son application en balayant l'écran avec son doigt, en étant guidée par une synthèse vocale qui énonce les éléments graphiques qui se trouvent sous son doigt. Le système Android de Google intègre justement ce type de lecteur d'écran (système Talkback) permettant de déplacer un focus de sélection de façon aléatoire sur l'ensemble des éléments visibles. Mais comme seuls les éléments visibles sont atteignables par ce moyen, le système prévoit un autre mécanisme (un geste multi points) pour déplacer la fenêtre visible. A ce premier inconvénient s'ajoute le fait que la précision du geste de pointage est dépendante de la taille et de l'agencement des éléments affichés. Pour pallier ces limitations, le système Talkback (à partir de la version Android 4.1) propose un autre mode de navigation dit « linéaire », permettant de déplacer pas à pas le focus. Dans ce mode, un geste rectiligne et rapide vers la droite déplace la sélection sur l'élément suivant selon un ordre de parcours prédéfini. Malheureusement, les deux modes d'interaction offerts, exploratoire et linéaire, cohabitent mal entre eux. Le premier problème vient du fait qu'il y a un risque pour l'utilisateur qui démarre sa tâche de sélection avec le mode linéaire de basculer dans le mode exploratoire, provoquant alors un changement inopportun du focus, qui saute subitement sur un des éléments de l'interface au lieu de suivre le parcours défini. Cela se produit lorsque le geste n'est pas suffisamment rapide au démarrage. Le deuxième problème vient du fait qu'il n'est pas possible de combiner ces deux modes au sein d'un même geste, en passant par exemple d'un mode de navigation par positionnement absolu à un mode de navigation par positionnement relatif. En effet, si l'utilisateur a d'abord utilisé le pointage absolu pour positionner approximativement le focus, il ne peut pas, sans lever le doigt, basculer dans le mode linéaire (i.e. séquentiel) afin d'ajuster sa sélection. Pour utiliser le mode linéaire il doit nécessairement interrompre son geste, ce qui nuit à la fluidité et à la cohérence de l'interaction. Ce mode séquentiel étant basé sur un critère de vitesse, un troisième inconvénient vient du fait qu'il ne permet pas non plus de faire défiler plusieurs éléments à la fois. En effet, pour y parvenir, il faudrait alors prendre en compte la distance parcourue par le doigt. Cela supposerait que le geste puisse démarrer lentement afin de pouvoir contrôler le défilement du focus sur les éléments graphiques. Or, si le geste démarre lentement, c'est le mode d'interaction exploratoire qui prime sur le mode séquentiel. Enfin, un lecteur d'écran ne fait que lire les éléments présentés à l'écran, leur agencement, leurs propriétés visuelles, sont autant de codes graphiques dont l'utilisateur n'a que faire. En effet, sa principale préoccupation est de faire correspondre sa tâche avec l'arbre logique des commandes disponibles. Ces mêmes limitations existent dans la surcouche d'accessibilité VoiceOver de l'iPhone. En conclusion, il existe aujourd'hui d'une part des interfaces dédiées aux personnes voyantes, qui peuvent porter leur attention visuelle sur l'écran de leur terminal et disposent d'une bonne habileté gestuelle, d'autre part, des interfaces dédiées aux malvoyants, mais qui n'offrent pas d'interaction cohérente, puisqu'elles nécessitent une succession de gestes interprétés selon des modes d'interaction distincts, source d'erreur et de confusion pour l'utilisateur. Nous expliquons ci-après comment notre solution répond à ces différents problèmes.

Principes de navigation retenus

Notre solution repose sur (1) un menu linéaire hiérarchique (2) une logique séquentielle impliquant un focus de sélection, (3) quatre commandes gestuelles de navigation : valider (*In*), retour au menu supérieur (*Out*), item précédent (*Up*) et suivant (*Down*). Les gestes correspondant aux deux commandes, *In* et *Out*, exploitent l'axe horizontal. Ceux des commandes *Up* et *Down* exploitent l'axe vertical.

Pour répondre au problème de la navigation en aveugle dans les interfaces, notre technique retient le mode séquentiel, mais sans la contrainte de vitesse et en utilisant l'ordonnée relative pour contrôler de façon continue le défilement du focus. On introduit alors le paramètre de *distance d'activation*, noté DA_y. Ce paramètre représente la distance minimale que le doigt, glissant suivant l'axe vertical, doit parcourir pour faire

avancer (ou reculer selon le sens du mouvement) le focus d'un élément, éventuellement de façon répétée. Ainsi, pour une même amplitude de mouvement, si ce paramètre est petit le nombre d'items de liste survolés par le focus sera grand, et vice versa, si ce paramètre est grand le nombre d'items survolés sera faible. A noter que `DA_y` fixe la vitesse de défilement logique du focus indépendamment de la représentation graphique. Il s'agit d'un paramètre de sensibilité de la navigation en largeur dans l'arborescence.

L'axe horizontal est quant à lui réservé à la navigation en profondeur dans la hiérarchie du menu, à la manière du *zoom sémantique* [8], en se basant sur l'abscisse relatif. On introduit de façon similaire le paramètre de *distance d'activation* `DA_x`. Ainsi, lors d'un geste rectiligne vers la droite, l'item sélectionné (possédant le focus) est activé (commande *In*) dès lors que cette distance d'activation est franchie. Inversement, lors d'un geste horizontal vers la gauche, le franchissement de cette même distance provoque la remontée au menu supérieur (commande *Out*).

Enfin, le geste *tap* (i.e. appui bref exécuté n'importe où) est utilisé pour obtenir un retour d'information vocal sur l'état de l'interface, en particulier sur le nom de l'item focusé. Tous ces gestes peuvent donc démarrer n'importe où sur l'écran. Cette approche permet aussi d'accéder aux éléments graphiques qui ne sont pas directement affichés dans la fenêtre visible du menu. Pour ce faire, lorsque le focus arrive en position haute ou basse de la fenêtre visible, il reste bloqué et c'est la liste qui défile. Que ce soit le focus qui bouge ou bien la liste, la loi de défilement reste la même.

Cette gestuelle étant basée sur le *positionnement relatif* (et non pas absolu), la contrainte de coordination visio-motrice est relâchée. De plus, comme nous n'utilisons que deux axes, la demande de précision sur l'orientation du geste est faible. Cela en fait une technique de menu adaptée aux situations de faible contrôle visuel ou moteur. Par ailleurs, en associant au déplacement du focus une synthèse vocale, cela en fait aussi une technique appropriée pour la manipulation en aveugle. Dans ce cas l'item vocalisé est celui qui possède le focus, la position du doigt pouvant être quelconque. A noter que cette technique se distingue du menu EarPod [21] qui associe chaque item à un secteur angulaire (une zone) et dont la manipulation en aveugle repose sur une exploration spatiale couplée à un positionnement absolu.

Pour répondre au besoin des utilisateurs voyants de pointer directement sur les éléments visibles du menu, notre technique prend également en compte le *positionnement absolu*. Pour ce faire, sur détection d'un appui statique pendant un temps prédéterminé de l'ordre de 300 millisecondes, l'élément cible est sélectionné, le focus se positionnant sous le doigt de l'utilisateur. Si l'appui se prolonge, l'activation a lieu après une durée prédéfinie, sinon le focus reste simplement à sa nouvelle position. A tout moment l'utilisateur peut décider d'activer sa sélection via un glisser vers la droite (ou *swipe*) effectué n'importe où sur l'écran, comme pour la manipulation en aveugle. Cette possibilité d'utiliser le pointage direct pour la sélection et/ou la validation est facultative, elle peut être désactivée dans le cas notamment d'un profil non-voyant, ceci afin de prévenir les erreurs de manipulation.

Enfin, pour tendre vers une interaction unifiée et souple, la technique MenuDfA permet de combiner de façon intuitive les deux types de positionnements, *absolu et relatif*. En effet, une fois la sélection effectuée par pointage direct, le système d'interprétation du geste bascule automatiquement en positionnement relatif (bidimensionnel) de sorte que l'interaction puisse se poursuivre en mode séquentiel. Cela permet, par exemple, à l'utilisateur voyant d'ajuster sa sélection en cas de pointage imprécis, mais aussi de valider dans la foulée sa sélection par un mouvement vers la droite. Autrement dit, après une phase de sélection par pointage direct, l'utilisateur garde le contrôle sur l'élément sélectionné, ceci avec les mêmes paramètres de sensibilité `DA_x` et `Da_y` que dans le mode séquentiel. Avec cette technique, la tolérance vis à vis de la précision du geste est indépendante de la taille et de l'agencement spatial des éléments graphiques. Cette propriété d'indépendance à la représentation graphique est d'un grand intérêt pour l'adaptation de l'interface aux besoins de l'utilisateur (par exemple une augmentation de la taille des polices) et également aux caractéristiques de son dispositif mobile (résolution, taille, facteur de forme).

Détection du mouvement et de l'orientation

Le principe de la détection du mouvement (de type positionnements relatifs) est basé comme nous l'avons vu sur les notions d'abscisse et d'ordonnée relatives. Le déclenchement des commandes de navigation verticale *Up* et *Down*, lié au déplacement du focus, dépend du paramètre de distance `DA_y`. L'algorithme simplifié est le suivant : soit `Ord`, l'ordonnée relative du geste. Au démarrage du geste : `Ord = 0`.

```
A chaque nouveau point de coordonnées
{x(n), y(n)} faire :
Dy = y(n) – y(n-1)    (supposé ici positif)
Si (Ord < DA_y)
Faire :
      Ord ← Ord + Dy
Sinon :
      Exécuter la commande Down
      Faire : Ord = 0
```

Pour la navigation en profondeur dans le menu hiérarchique (commandes *In* et *Out*), les principes sont les mêmes, mais avec une distance d'activation propre : `DA_x`. Mais au démarrage du geste, le système doit décider de l'orientation verticale ou horizontale du mouvement, ceci afin d'émettre la commande appropriée

(*In*, *Out*, *Down* ou *Up*). Dans le menu MenuDfA, cette détection se fait de manière robuste en définissant quatre régions de décision dans l'espace moteur, chacune associée à une direction cardinale, comme représentée Figure 2. Chaque région est un secteur délimité entre deux bissectrices, hormis la zone hachurée qui représente une zone de non décision liée aux deux paramètres de sensibilité : DA_x et DA_y. La trace partant de l'origine représente un exemple de trajectoire au doigt déclenchant deux fois successivement la commande *In* après avoir franchi la région d'indécision.

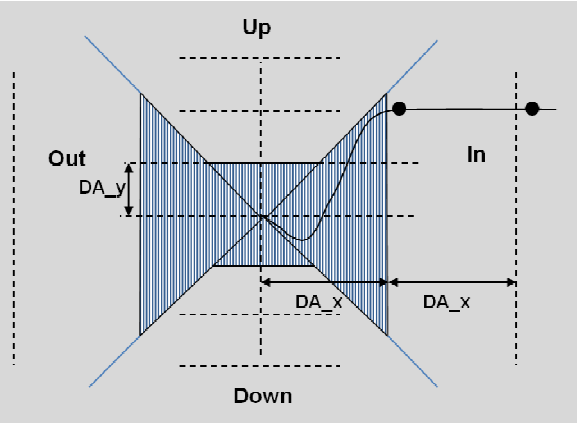

Figure 2: détection de l'orientation du geste au démarrage

Ces principes, intégrés dans le moteur de reconnaissance DGIL, sont complétés par un algorithme permettant de détecter de façon fiable un changement de direction du geste en cours de production. Il devient alors possible de reconnaitre des gestes complexes de type composition spatiale, comme indiqué dans le paragraphe suivant traitant de l'optimisation de l'interaction.

Optimisation de l'interaction

Dans le cas des listes longues et lorsque l'élément cible n'est pas visible, la navigation séquentielle en largeur dans l'arborescence peut s'avérer fastidieuse car elle impose de parcourir les éléments un par un. Cet inconvénient est atténué par le fait que l'utilisateur peut utiliser un geste continu pour faire défiler plusieurs items en une seule action. De plus, la sensibilité du défilement, déterminée par le paramètre DA_y, peut être ajustée en fonction du type de liste ou des préférences de l'utilisateur. Mais pour gagner encore en efficience, notre technique intègre un moteur d'inertie permettant de simuler un mouvement physique et de prolonger ainsi artificiellement le geste de l'utilisateur. Cette propriété permet alors d'atteindre sans effort un élément de fin de liste. Par ailleurs, comme nous l'avons vu, la technique MenuDfA propose une gestuelle simple basée sur des gestes de pointage ou des gestes rectilignes verticaux ou horizontaux (non localisés). En particulier, un utilisateur novice peut parcourir l'arborescence du menu par une série de gestes directionnels simples, à la manière des Multi-Stroke Menus de Zhao [20]. Précisons toutefois que la comparaison de notre technique avec les menus circulaires [10, 2] n'est valable que partiellement. En effet, pour ce type de menus, la direction du geste est liée à la disposition spatiale des items par rapport au centre du menu, ce qui en limite le vocabulaire gestuel. Dans notre technique de menus, la direction du geste dépend uniquement du type d'exploration logique, en largeur ou en profondeur dans l'arborescence du menu. Et quelle que soit la largeur du menu courant, quatre commandes (réversibles) et deux axes orthogonaux suffisent.

Enfin, comme nous l'avons évoqué, notre technique possède un geste performant pour traverser rapidement la hiérarchie du menu. En effet, le MenuDfA est capable d'interpréter des *gestes composés* spatialement, effectués depuis n'importe quel sous-menu. Le geste est alors formé d'une succession de segments orientés et d'inflexions. Ainsi, un utilisateur pourra par exemple séquentiellement effectuer une sélection (segment vertical), activer un item de catégorie (segment horizontal) qui le fera descendre dans un sous-menu, sélectionner (vertical) puis valider une case à cocher (horizontal vers la droite) et remonter dans le menu parent (horizontal vers la gauche). Cet enchainement de commandes pourra donc être réalisé en un seul geste composé, sans jamais avoir à lever son doigt (voir Figure 3). En conclusion, si ce geste ne constitue pas à proprement parler un raccourci gestuel, comme proposé dans les modes experts des techniques [3,4,10,1], il offre néanmoins un style de navigation rapide et fluide.

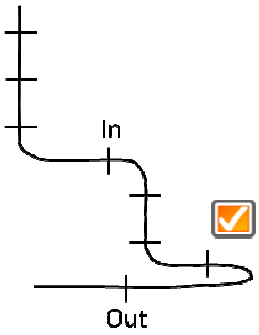

Figure 3 : geste composé

Notons que l'utilisateur contrôle à tout moment la complexité de son geste. Il peut atteindre son but en plusieurs gestes simples ou en un seul geste composé si l'espace écran le permet. Qui plus est, il est adapté à un usage du smartphone manipulé au pouce d'une seule main.

Contrôle utilisateur et guidage de l'interaction

Le contrôle utilisateur sur le traitement de ses actions et le guidage de son action par le système, sont des aspects cruciaux vis-à-vis de l'utilisabilité. D'après les critères ergonomiques en vigueur [5] l'utilisateur doit être guidé dans son interaction par des *retours immédiats* sur ses actions et aussi des incitations à l'action. Il doit aussi bénéficier de mécanismes de protection vis-à-vis des erreurs de manipulation. Or, la nature discontinue du déplacement du focus sautant d'un élément à l'autre

alors que le doigt glisse de façon continue sur la surface n'est pas entièrement satisfaisante vis-à-vis de ces critères. C'est pourquoi, nous proposons de modifier la loi en escalier de déplacement du focus. Cette loi définit la fonction de transfert entre l'espace moteur et l'espace d'affichage. La loi modifiée est représentée par le graphe de la Figure 4. Sur ce graphe, la courbe rectiligne (en trait pointillé) représente l'évolution du déplacement du doigt en fonction du temps, sous l'hypothèse d'un mouvement rectiligne uniforme et vertical, comme illustré sur la copie d'écran de smartphone à droite. Son ordonnée augmente donc de façon linéaire avec le temps. La courbe discontinue (en trait continu) est celle du déplacement du focus en fonction du temps.

On constate que la transition du focus, lors d'un changement de sélection, est maintenant constituée de trois phases. La première phase (❶), dite « linéaire », correspond à un déplacement du focus proportionnellement à la distance parcourue par le doigt. La deuxième phase (❷), produite à l'instant `t1` (puis `t2`), correspond à un saut instantané du focus (Δs sur la figure) se repositionnant sur le nouvel item sélectionné. La troisième phase (❸) correspond à un blocage du focus sur le nouvel item, laissant le doigt continuer son mouvement sur une certaine distance (ΔD) avant de repartir en phase 1 (instant `t1'`). L'implémentation de ce mécanisme s'appuie sur un paramètre variable, `p`, fourni par DGIL au cours du geste. Ce dernier correspond au pourcentage de distance parcourue par le doigt par rapport à la distance d'activation fixée, soit, en gardant les notations précédentes :

```
p(Ord) = (Ord/DA_y) * 100   (1)
```

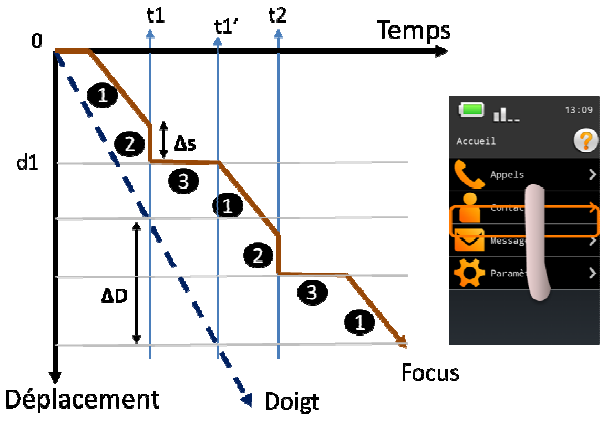

Figure 4: loi de déplacement du focus

Au niveau du composant d'interface, la loi implémentée du déplacement du focus en fonction de `p` est la suivante :

```
Dep(p) = (p*alpha) * Hc /100   (2)
```

où `Hc` est la hauteur de la cellule de l'item courant. Et `alpha` (environ 0.7) est un paramètre qui fixe la part de la phase linéaire vis à vis de la phase de saut (phase 2). Cette loi simple tient compte du paramètre de sensibilité défini au niveau de l'espace moteur. Elle tient compte également de la hauteur des items afin de produire un même motif de déplacement quelque soit l'échelle. De fait, elle fonctionne encore avec des éléments de hauteur variable. En réalité, la loi (1) est modifiée pour pouvoir mettre en œuvre la phase 3, ceci de façon transparente pour l'interface, la loi (2) restant inchangée. Ainsi, le paramètre `p` est forcé à zéro à la fin de la phase 2 et durant toute la phase 3, afin de créer l'effet de blocage. A noter que celui-ci correspond à un effet pseudo-haptique qui participe à l'amélioration du contrôle utilisateur. Plus globalement, ces trois phases jouent un rôle important au niveau de l'utilisabilité de la technique et aussi au niveau du critère de satisfaction. Dans cette hypothèse, chaque phase à un but ergonomique précis :

❶ : procurer un retour d'information immédiat sur l'action de l'utilisateur lors de la manipulation et procurer de la fluidité lorsque l'inertie est enclenchée.

❷ : marquer le changement de sélection, effet pouvant être accompagné d'un retour sonore ou vibratoire.

❸ : améliorer la précision gestuelle (l'objet est plus longtemps sélectionné) et sécuriser la manipulation lors d'un *geste composé*.

Soulignons que cette fonction de transfert est flexible. Elle peut être ajustée en fonction d'un type d'interface, d'un profil utilisateur ou d'une situation d'usage.

Retours d'information visuel et vibratoire

L'utilisation d'une technique de manipulation gestuelle tactile « déspatialisée », où l'espace visuel est séparé de l'espace moteur, est parfaite pour un utilisateur non-voyant mais n'est pas habituelle chez un utilisateur voyant. Aussi, pour l'aider à apprendre cette nouvelle manière de naviguer dans les menus, nous avons introduit deux types d'effets visuels accompagnés chacun d'un retour haptique. Le premier concerne la commande de validation (*In*). Il consiste en un décalage horizontal vers la droite de la cellule focusée lié au glissement du doigt vers la droite (voir Figure 6). Ce couplage fort entre le focus et le doigt fait comprendre à l'utilisateur qu'il est en train de manipuler l'item sélectionné même s'il ne le pointe pas directement. Il perçoit ainsi le lien de cause à effet entre son geste, pouvant démarrer d'une position spatiale quelconque, et la commande d'activation liée à l'item courant, et seulement à celui-ci. Ce *feedback* visuel continu lui permet aussi de bien contrôler son action et le cas échéant de s'en échapper par un mouvement dans une autre direction. Il est renforcé par une vibration discrète déclenchée lorsque la distance d'activation est atteinte. Cette dernière permet à l'utilisateur de percevoir haptiquement la distance d'activation et par conséquent de sécuriser la navigation gestuelle continue. La Figure 5 représente la phase de sélection et la Figure 6, celle de validation, lors d'un geste composé.

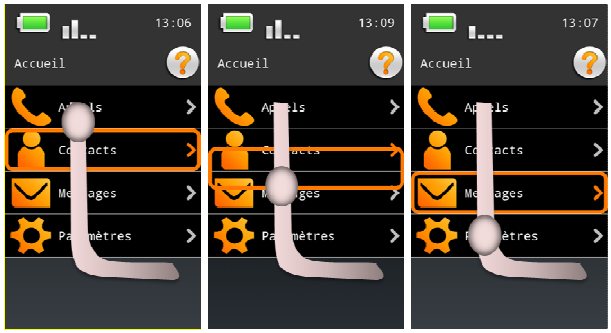

Figure 5: effet visuel sur un geste composé (trace du doigt en rose pâle) en phase de sélection

Figure 6 : effet visuel sur un geste composé (trace du doigt en rose pâle) en phase de validation

De plus, nous avons mentionné plus haut que l'activation d'un élément du menu pouvait être réalisée de deux façons : par un geste de gauche à droite sans contrainte de positionnement spatial, ou bien par un appui prolongé effectué directement sur l'élément cible. Dans ce deuxième cas, nous avons créé un effet visuel identique de décalage vers la droite. Cela correspond à associer un même type de retour visuel à une même commande qui peut être effectuée par deux gestes différents.

Le deuxième effet concerne la commande de retour au menu supérieur (*Out*). Il consiste en un décalage horizontal vers la droite de la fenêtre du menu courant (ou la page d'écran entière selon le cas) lié au glissement du doigt vers la gauche. Ce décalage laisse apparaître progressivement le menu parent avec son focus bien positionné sur l'item correspondant. Cet effet facilite la compréhension par l'utilisateur du chemin de navigation dans la hiérarchie. Il permet aussi de mieux contrôler visuellement la distance parcourue par le doigt, conjointement avec une vibration déclenchée lorsque la distance d'activation est franchie. Guidé par cet effet visuel et par la vibration, l'utilisateur peut de façon maitrisée remonter en un seul geste de plusieurs niveaux dans la hiérarchie du menu. A noter qu'au départ le doigt entraine la page par le biais du moteur DGIL, puis une fois le seuil de distance atteint, l'animation se termine de manière automatique. La figure 7 illustre cette transition visuelle dans le cas du retour à la liste précédente depuis le contenu d'un courriel.

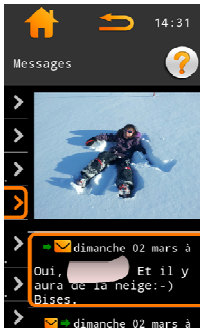

Figure 7 : effet visuel sur une commande de retour

Interaction gestuelle fine

Comme nous l'avons indiqué, l'interaction gestuelle de la technique MenuDfA repose sur DGIL [13]. Ce framework d'interaction s'inspire des idées de Bill Buxton [6] qui proposait déjà en 1986 de structurer le dialogue gestuel homme-machine en concevant le geste kinesthésique comme une phrase constituée d'unités significatives permettant d'organiser naturellement la succession des opérations élémentaires lors d'une tâche d'interaction. Cette phrase devait être articulée à l'aide d'un seul geste caractérisé par une tension musculaire constante jusqu'au relâchement final. Ces idées, bien que déjà anciennes, appellent encore aujourd'hui à une meilleure segmentation physique et logique du geste tactile afin de définir les blocs élémentaires d'un langage d'interaction gestuelle nécessaire pour concevoir les NUIs[2]. C'est dans cette perspective que nous avons conçu DGIL et créé un langage d'interaction formé de 36 primitives gestuelles. La philosophie de DGIL réside dans une approche unifiée du geste, où le moindre événement tactile (de type mono point) capté par la dalle sensible est considéré *a priori* comme un geste ou une partie d'un geste intentionnel. Ainsi, un simple « tap » est un geste au même titre qu'un geste symbolique représentant une forme complexe. Dans cette approche, les caractéristiques physiques du geste sont analysées, y compris le temps, ce qui permet de prendre en compte sa dynamique. La finalité de DGIL est alors de fournir à l'interface utilisateur, et en temps réel, une description syntaxique fine du geste sous la forme d'une séquence de primitives gestuelles accompagnée de paramètres. Ces primitives sont autant d'événements significatifs que l'on peut connecter aux commandes et *feedbacks* de l'interface. Par exemple, dans le cas du geste composé illustré Figure 5 et Figure 6, la séquence produite est la suivante :

PRESS → START_MOVE → DRAG →
SEQ_EARLY_MOVE_DOWN(16,27,42,59,75,85) →
SEQ_MOVE_DOWN → DRAG →
SEQ_EARLY_MOVE_DOWN(2,14,33,42,90) →
SEQ_MOVE_DOWN → DRAG →
SEQ_EARLY_MOVE_DOWN(10,24,42,63) →
SEQ_CANCEL_EARLY_MOVE → DRAG →

[2] Natural User Interfaces.

SEQ_EARLY_MOVE_RIGHT(38,62,88) → **SEQ_MOVE_RIGHT** → END_DRAG → END_GESTURE

La gestion fine du focus en 3 phases est assurée par la primitive SEQ_EARLY_MOVE_DOWN dotée du paramètre p (Cf. la loi 1). Les commandes *Down* sont déclenchées par la primitive SEQ_MOVE_DOWN. L'effet de décalage horizontal de la cellule de l'item courant est produit grâce à la primitive SEQ_EARLY_MOVE_RIGHT, dotée également de son paramètre. La primitive SEQ_CANCEL_EARLY_MOVE permet de repositionner le focus sur l'item courant au moment du changement de direction. Enfin, la primitive SEQ_MOVE_RIGHT déclenche l'activation de la sélection.

DGIL est ici configuré dans un mode spécifique dédié à la navigation séquentielle par positionnement relatif. D'autres modes d'interaction sont possibles comme celui utilisé dans la fonction Gestes d'Orange [13] permettant de combiner les gestes de manipulation directe avec les gestes symboliques.

REALISATION

La réalisation du MenuDfA, en tant que composant d'interface pour plateforme Android OS, repose sur plusieurs bibliothèques logicielles écrites en Java ou C++. L'architecture logicielle est basée sur un modèle MVC (Modèle, Vue, Contrôleur) et utilise une programmation événementielle induite par le framework DGIL et dont la mise en œuvre est facilitée par le *micro-framework* AEvent[3]. Ce dernier permet de définir de manière souple des associations entre des événements ou cascades d'événements et des commandes. On crée alors des objets *handler* qui regroupent une collection d'abonnements «événement-commande(s)».

En particulier, le *handler* de DGIL a pour rôle de transformer les primitives du moteur en événements AEvent, en prenant en compte le style de navigation du menu et le contexte d'interaction. Il devient alors possible de concevoir des événements gestuels plus complexes faisant appel à une combinaison ou à une cascade d'événements (issus ou non de DGIL) et de commandes. C'est le cas notamment de *l'événement composite* « Long press tolerant right » qui permet d'exécuter la commande d'activation de plusieurs manières afin d'assouplir les contraintes sur le geste. Il autorise la combinaison souple de l'appui statique et du glissement vers la droite. En effet, après avoir effectué une sélection au moyen d'un appui court (*short press*), l'utilisateur dispose de trois actions pour terminer sa commande de validation. Soit il prolonge son appui statique jusqu'au déclenchement de la validation ; le critère est alors temporel (*time out*). Soit il valide sans attendre à l'aide d'un glisser vers la droite (*move right*); dans ce cas, c'est le franchissement de la distance d'activation qui déclenche la commande. Soit il démarre un glisser vers la droite mais pas suffisant pour franchir le seuil de distance, auquel cas c'est le critère de durée qui prend le pas sur celui de distance et qui déclenche la commande.

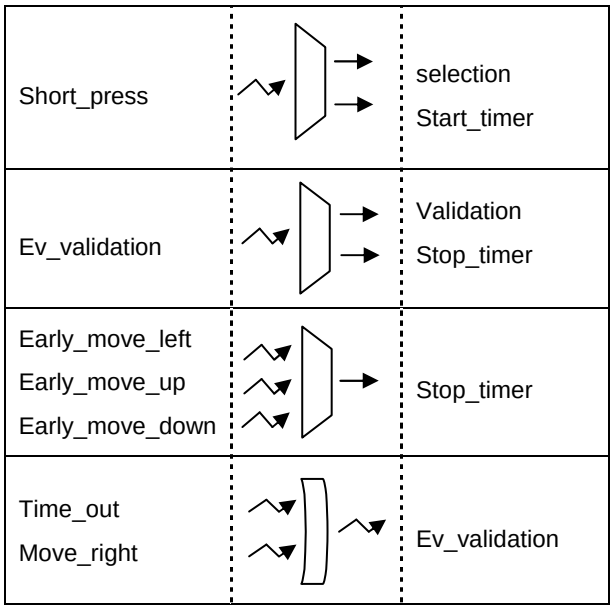

Tableau 1: description de l'événement composite associé à la commande de validation

Le Tableau 1 décrit de façon plus formelle cet *événement composite*. Dans ce schéma, l'événement Short_press est connecté simultanément aux commandes Selection et Start_timer, l'événement Ev_validation aux commandes Validation et Stop_timer. Par ailleurs, la commande Stop_timer est aussi abonnée aux trois événements : Early_move_left, Early_move_up et Early_move_down, tous issus de DGIL. Cela signifie qu'un déplacement du doigt dans une direction autre que celle vers la droite, a pour effet de mettre fin au mécanisme temporel de déclenchement de la validation. Il s'agit d'une procédure d'échappement. Enfin, les événements Time_out et Move_right (issu de DGIL) sont mis en concurrence pour former l'événement Ev_validation. Cet événement composite est donc implémenté par le biais du *micro-framework* AEvent, dont l'association avec DGIL apporte une solution d'ingénierie flexible et bien adaptée au design d'interaction gestuelle.

Au sein du modèle MVC, le Contrôleur supervise l'exécution des commandes de navigation, en relation avec la Vue et le Modèle. Ce dernier, gère la structure de données du menu hiérarchique qui est chargée au démarrage de l'application. La partie statique du menu peut être décrite en langage XML. Concernant la Vue du modèle MVC, elle est constituée principalement de deux composants graphiques basés sur l'API Android (niveau 17), à savoir : la *DfAList* et le *DfAPager*. Le premier composant, la *DfAList*, gère une liste à accès séquentiel, basée sur un focus de sélection. Cette liste peut être circulaire ou non, posséder des items sélectionnables ou non sélectionnables (le focus passant alors par-dessus).

[3] Stéphane Coutant, Orange Labs

La nature de l'item est fixée par l'application. Selon son type (nœud ou feuille), l'item peut être un item de catégorie (i.e. un sous-menu), ou un item de type case à cocher ou bouton radio, une commande, une image, voire un hyperlien. Cette liste ayant des propriétés spécifiques de manipulation indirecte via un focus, son implémentation a nécessité un développement original ne faisant pas appel à la classe Java *ListView* de l'API standard. Le second composant, *DfAPager*, gère l'enchainement des menus *DfAList* le long du chemin de navigation. Il présente le menu courant dans une page constituant la fenêtre principale de l'écran. Par ailleurs, lors d'un changement de niveau hiérarchique, il fait apparaître les pages adjacentes contenant le menu parent ou fils selon le sens de la navigation. Il a été conçu pour garantir à la fois la continuité gestuelle et visuelle de l'interaction entre les niveaux hiérarchiques. Il permet notamment de mettre en œuvre l'effet de décalage de page associé à la commande de retour (Cf. Figure 7).

Concernant le module de reconnaissance DGIL, il a été développé en C++ mais est doté d'une API Java dédiée à la plateforme Android. Son API fournit un ensemble de méthodes d'objets (de type *Event Listener* en Java) associées aux différentes primitives gestuelles, éventuellement accompagnées de paramètres. La partie algorithmique, optimisée pour une architecture de processeur ARM, est fournie sous la forme d'une bibliothèque dynamique C++. Cette partie est exécutée en natif sur la plateforme Android. Précisons que ce module est à un niveau de développement industriel depuis 2011 et qu'il fait régulièrement l'objet d'évolutions afin de traiter de nouveaux types de gestes, comme les gestes composés et gestes dotés d'une inertie.

Prototypage et tests utilisateurs

Une première validation du principe de navigation séquentielle par positionnement relatif a été réalisée en 2012, au travers de la méthode de saisie de texte en aveugle oNavTouch [14]. Ce prototype a été soumis à un test utilisateur incluant huit personnes mal et non voyantes. Il a conclu à une bonne utilisabilité de la méthode au regard du système VoiceOver d'Apple.

Un deuxième prototype fonctionnel a été réalisé en 2013 basé sur la technique MenuDfA, ceci dans le cadre de l'application One-button Phone[4]. Cette application a été conçue pour permettre à des personnes atteintes de déficiences motrices de manipuler un smartphone et d'accéder à ses fonctions principales. Elle proposait trois profils d'interaction dont un mettant en œuvre un défilement automatique du focus. Sa conception ergonomique a été guidée par des travaux antérieurs sur le handicap moteur. Ce prototype nous a permis d'obtenir des premiers éléments de faisabilité du MenuDfA vis-à-vis d'une logique de « conception pour tous ». Enfin, concernant les principes décrits dans cet article, ils ont été mis en œuvre au sein d'une nouvelle version plus aboutie du MenuDfA, en cours d'industrialisation et destinée à une application de téléphonie mobile pour novices et personnes âgées.

CONCLUSION

Nous avons présenté dans cet article une nouvelle technique de menu hiérarchique linéaire qui relâche les contraintes visuo-motrices tout en offrant de nombreux paramètres de configuration. Ces propriétés permettent de prendre en compte des utilisateurs ayant des difficultés motrices ou une acuité visuelle limitée, ainsi que certaines situations d'usage où l'attention visuelle est partagée. De plus, notre technique ne suppose pas de limitation en largeur au sein de l'arborescence, ce qui la rend applicable à de nombreux cas d'usages, notamment dans le contexte des applications pour terminaux mobiles.

Dans cette approche, des gestes de positionnements relatifs sont introduits en même temps que des retours visuels innovants afin de guider l'utilisateur dans sa tâche de navigation. On introduit également la possibilité d'exécuter des compositions gestuelles spatiales, ainsi que celle de combiner des gestes de pointage avec des gestes de positionnements relatifs. Cette gestuelle « déspatialisée » apporte une interaction continue, fluide et sécurisante. Elle s'applique de façon homogène dans toute la hiérarchie du menu.

Cette version du MenuDfA est plus spécialement destinée à des utilisateurs voyants mais semble avoir tout le potentiel pour adresser d'autres profils utilisateur (handicap visuel, moteur, cognitif) et tendre vers une solution d'accessibilité universelle. Cela fait partie de nos perspectives et impliquera de mener des évaluations expérimentales plus poussées.

Nous envisageons aussi de réaliser une déclinaison du MenuDfA adaptée aux dispositifs de type tablette tactile. L'idée serait de présenter à l'utilisateur simultanément deux niveaux hiérarchiques : le menu courant et le sous-menu associé à la sélection, ceci afin de mieux exploiter l'espace d'affichage et également d'enrichir son contexte d'interaction.

REMERCIEMENTS

Les auteurs remercient S. Coutant pour la réalisation des premiers prototypes, C. Maldivi pour les évolutions de DGIL, S. Zijp-Rouzier pour sa contribution à la conception ergonomique, M. Gimenes et F. Darbas pour les développements logiciels du composant d'interface. Ils remercient également E. Froc pour avoir pris le risque de l'utiliser au sein d'une application opérationnelle, ainsi que D. Ponge, P.Y. Folens et S. Roux.

[4] http://www.youtube.com/watch?v=xDXvLqes3-E

BIBLIOGRAPHIE

1. Bailly G. Techniques de menus : caractérisation, conception et évaluation. Thèse en informatique, Univ. J. Fourier, mars 1992.
2. Bailly G., Lecolinet E. & Nigay L. Quinze Ans de Recherche sur les Menus: Critères et Propriétés des Techniques de Menus. Actes d'IHM'07, Nov. 2007, Paris, ACM Press.
3. Bailly G., Roudaut A., Lecolinet E. & Nigay L. Menus leaf : enrichir les menus linéaires par des gestes. Actes d'IHM '08, ACM (2008).
4. Bailly G., Lecolinet E., & Nigay L. *Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization.* In Proc. AVI '08, ACM (2008).
5. Bastien J.M.C., Scapin D. Ergonomic Criteria for the Evaluation of Human-Computer interfaces. Rapport de Recherche de l'INRIA (1993) : http://hal.inria.fr/docs/00/07/00/12/PDF/RT-0156.pdf
6. Buxton W. *Chunking and phrasing and the design of human-computer dialogues.* Proc. of the IFIP World Computer Congress (1986), 475-480.
7. Clerc I. La démarche de rédaction. Québec, Éditions Nota Bene. (2000).
8. Lecolinet E. & Nguyen D. Représentation focus+contexte de listes hiérarchiques zoomables. Actes d'IHM'06, ACM Press.
9. Kobayashi M. & Igarashi T. *Considering the direction of cursor movement for efficient traversal of cascading menus*. In Proc. UIST '03, ACM (2003).
10. Kurtenbach G. & Buxton W. *User Learning and Performance with Marking Menus*. Proc. of CHI '94, ACM (1994), p. 258-264.
11. Nancel M., Huot S. & Beaudouin-Lafon M. Un espace de conception fondé sur une analyse morphologique des techniques de menu. Actes d'IHM'09, Oct. 2009, Grenoble, ACM Press.
12. Norman D. A. (1988). *The Design of Everyday Things.* Revised and Expanded Edition. New York: Basic Books. London: MIT Press (UK edition) (2013).
13. Petit E. & Maldivi C. Démonstrations autour d'un nouveau framework d'interaction gestuelle tactile. IHM 2013, session démonstrations : http://hal.inria.fr/docs/00/87/95/37/PDF/05-DGIL-IHM2013_DGIL_demo_final.pdf
14. Petit E. & Ponge D. oNavTouch : méthode de saisie de texte sur téléphone mobile tactile adaptée au contexte du handicap visuel. Forum sur l'Interaction Tactile et Gestuelle (FITG), Tourcoing, 2012. http://fitg12.lille.inria.fr/exposes/petit-ponge/index.html
15. Pook S., Lecolinet E., Vaysseix G. & Barillot E. *Control menus: excecution and control in a single interactor*. In proc. *CHI '00 Extended Abstracts*, ACM (2000), p. 263-264.
16. Roudaut A., Lecolinet E. & Guiard Y. *MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb*. In Proc. CHI '09, ACM (2009).
17. Shneiderman B. *Direct Manipulation: A Step Beyond Programming Languages*. Computer, Vol. 16, No. 8 (1983), p. 57-69.
18. Stephanidis C. Designing User Interfaces for All. Proceedings of the Center On Disabilities Technology And Persons With Disabilities Conference, 1998. http://www.csun.edu/cod/conf/1998/proceedings/csun98_032.htm
19. Vogel D. & Baudisch P. *Shift: a technique for operating pen-based interfaces using touch*. In proc. CHI '07, ACM (2007).
20. Zhao S. & Balakrishnan R. *Simple vs. compound mark hierarchical marking menus*. In Proc. UIST '04, ACM (2004), p. 33-42.
21. Zhao S., Dragicevic P., Chignell M., Balakrishnan R., Baudisch P. *earPod: Eyes-free Selection using Touch Input and Reactive Audio Feedback*, In Proc. CHI'07, ACM (2007).