MenuDfA : vers une navigation gestuelle tactile conçue pour tous

Eric Petit, Denis Chêne

MenuDfA : vers une navigation gestuelle tactile conçue pour tous

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

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

RÉSUMÉ
Cet article présente une nouvelle technique de navigation au doigt dans une interface tactile conçue dans une approche de conception pour tous. Cette technique repose sur (1) une arborescence logique de navigation sous la forme d'un menu linéaire hiérarchique, (2) une logique séquentielle de navigation basée sur la manipulation d'un focus de sélection, (3) des gestes de "glissé" permettant la manipulation continue. Contrairement aux interfaces tactiles standards, l'interaction ne repose pas ici sur du positionnement absolu (ou pointage), mais sur du positionnement relatif, ce qui permet de relâcher la contrainte de coordination visuo-motrice. Elle est donc bien adaptée aux petits écrans et en cas d'attention visuelle partagée. De plus, en séparant les opérations de sélection et de validation elle conduit à une manipulation sûre compatible avec une utilisation en aveugle. Enfin, l'introduction de gestes continus et/ou composés conduit à une interaction fluide.

Mots Clés
Navigation gestuelle tactile, manipulation continue, positionnement relatif, technique de menus.

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.

NB : Cet article a été soumis à la conférence IHM 2015, mais rejeté au motif qu'il manquait une validation expérimentale. Néanmoins, la technique a été jugée convaincante et l'article motivé.

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, onglets, 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'ajoutent 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 aux situations à faible contrôle visuel et/ou faible contrôle moteur. Ces travaux sont issus de deux axes de recherche. Le premier axe porte sur la reconnaissance de gestes tactiles, le deuxième, sur la prise en compte du handicap visuel et moteur dans l'interaction homme-machine. Ce second axe a guidé nos choix de conception ergonomique, en particulier le choix des gestes et des règles de navigation. Soulignons que notre technique est applicable aux menus de type sélection de commandes dans un domaine fini [2, 11], mais aussi, plus largement, aux listes à nombre variable d'items, défilantes ou non, rémanentes ou non. En conséquence, elle n'impose pas que tous les éléments de la liste courante soient visibles et donc atteignables par pointage direct. Contrairement aux menus hiérarchiques habituels, notre approche n'impose pas de limitation quant à la largeur de l'arborescence, ce qui, d'après l'analyse MenUA [1], augmente grandement son *facteur d'applicabilité*. 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 de 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 qui nécessite d'être défini 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 et al. [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.

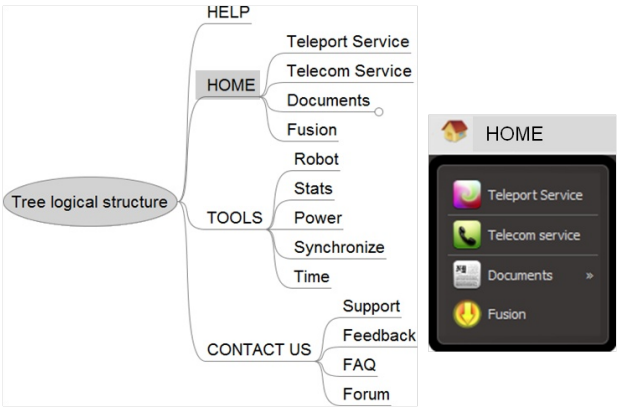

Figure 1: représentation logique des fonctions et représentation visuelle au sein d'un menu linéaire hiérarchique

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 déclinées 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 « déspatialisée » car fondée sur un parcours logique de navigation, (2) l'utilisateur ne manipule qu'un seul objet, le focus de sélection (3) et seul le menu courant nécessite d'être affiché. La logique séquentielle en œuvre est donc basée sur un focus de sélection que l'utilisateur manipule de façon continue 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 selon un mode séquentiel. 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, Icons, 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 se trouvant 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. À 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 démarrant 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 non-voyant 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, et d'autre part, des interfaces dédiées aux personnes malvoyantes, ou non-voyantes, 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 associé à une arborescence logique (2) une logique séquentielle impliquant un focus de sélection matérialisé par un cadre graphique rectangulaire, (3) quatre commandes gestuelles de navigation : valider (*In*), retour au menu supérieur (*Out*), item précédent (*Up*) et suivant (*Down*). Précisons que chaque commande peut être exécutée via un geste de « glissé », c'est-à-dire, en ayant seulement recours au *positionnement relatif*. Dans ce cas, les deux commandes, *In* et *Out*, exploitent l'axe horizontal, et les commandes *Up* et *Down* l'axe vertical.

Pour répondre au problème de la navigation en aveugle dans les interfaces nous introduisons la manipulation continue sans pointage. 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 inversement, 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 donc de façon similaire le paramètre de *distance d'activation* `DA_x`. Ainsi, lors d'un geste rectiligne vers la droite, l'item sélectionné (celui 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 alors 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), l'utilisateur peut démarrer son geste sans contrainte de coordination visuo-motrice. Par ailleurs, la précision sur la distance à parcourir peut être ajustée car liée aux paramètres `DA_x` et `Da_y`. De plus, comme les deux axes de glissement sont orthogonaux, la demande de précision sur l'orientation du geste est faible. En conséquence, cela fait du MenuDfA une technique de navigation tactile adaptée aux situations de faible contrôle visuel ou moteur. Enfin, en associant au déplacement du focus une synthèse vocale et des retours vibratoires, cela en fait aussi une technique appropriée pour la manipulation en aveugle. A noter que notre technique se distingue nettement du menu earPod [21] pour lequel chaque item du menu est associé 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, il peut aussi enclencher la validation (commande *In*). Dans ce cas, un appui localisé sur un item quelconque provoquera la sélection et ensuite la validation, comme dans une interface de pointage standard. 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 sécuriser la navigation.

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, l'utilisateur peut enchainer sur un geste de glissé, soit pour valider la sélection (geste vers la droite), soit pour l'ajuster en faisant glisser son doigt suivant l'axe vertical. Contrairement à TalkBack, après la phase de sélection par pointage, le MenuDfa bascule automatiquement en mode séquentiel et positionnement relatif, sans que l'utilisateur ait besoin de lever son doigt. L'utilisateur garde alors le contrôle sur l'élément sélectionné. Avec cette technique, la tolérance vis à vis de la précision du geste peut être gérée indépendamment de la taille et de l'agencement spatial des éléments graphiques. Cette propriété de faible dépendance à la représentation graphique est d'un grand intérêt pour l'adaptation de l'interface aux besoins de l'utilisateur ou aux caractéristiques de son dispositif d'interaction (taille, facteur de forme).

Détection du mouvement et de l'orientation

Le principe de la détection du mouvement du doigt est basé comme nous l'avons vu sur les notions d'abscisse et d'ordonnée relatives. Dans le cas de la navigation en largeur (au sein d'une liste présentée verticalement), le déclenchement des commandes de navigation *Up* et *Down*, lié au changement de sélection, 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. Sur la figure, 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.

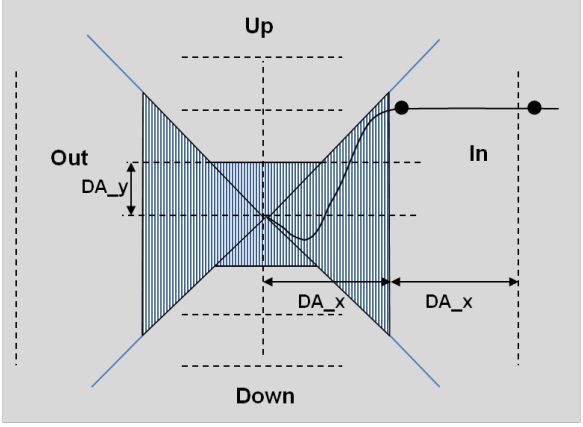

Figure 2: détection de l'orientation du geste au démarrage

Ces principes 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 gestuelle

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. Notre technique pallie cet inconvénient en permettant à l'utilisateur de faire défiler plusieurs items en une seule action, ceci avec un geste de glissé continu. Dans ce cas, la vitesse du défilement, fixée 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. L'utilisateur (novice) pourra aussi parcourir l'arborescence du menu par une série de gestes directionnels simples, à la manière des Multi-Stroke Menus de Zhao et al. [20]. Soulignons que dans le cas des menus circulaires [10, 2], 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. En effet, au-delà d'un certain nombre il devient difficile à la fois de représenter visuellement ces derniers mais également de différencier les gestes permettant de les atteindre. Dans notre approche, 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, MenuDfA propose en plus un geste performant permettant de traverser rapidement la hiérarchie du menu. Ce geste est une composition spatio-temporelle et continue de plusieurs gestes élémentaires de « glissé ». Il peut être effectué depuis n'importe quel menu ou sous-menu. Il est alors formé d'une succession de segments orientés et d'inflexions. Ainsi, un utilisateur peut par exemple séquentiellement effectuer une sélection d'un item de catégorie (segment vertical), puis poursuivre son geste sans lever son doigt en activant ce dernier (segment horizontal), puis descendre dans un sous-menu afin de sélectionner un autre item (segment vertical), puis valider une case à cocher (segment horizontal vers la droite) et enfin remonter au menu parent (segment 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.

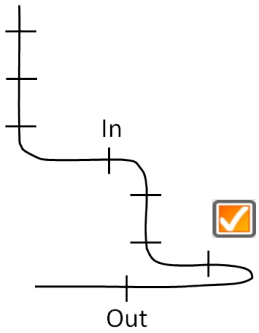

Figure 3 : geste composé

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, ainsi que 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 *incitations à l'action* et des *retours immédiats* sur ses actions. Il doit aussi bénéficier de mécanismes de protection vis-à-vis des erreurs de manipulation. Dans la technique MenuDfA l'utilisateur ne manipule qu'un seul objet, le focus de sélection. De fait, la fonction de transfert du couple doigt-focus, entre l'espace moteur et l'espace visuel, revêt une grande importance, en particulier lors d'un mouvement vertical. Dans ce cas, le déplacement du focus doit accompagner le doigt de façon précise et stable, ceci en assurant le plus possible la continuité du couplage entre la perception et l'action. Aussi, plutôt qu'une loi simple en escalier, nous avons introduit une loi en trois phases, 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. La transition du focus lors d'un changement de sélection est donc 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 le système. 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)
```

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.

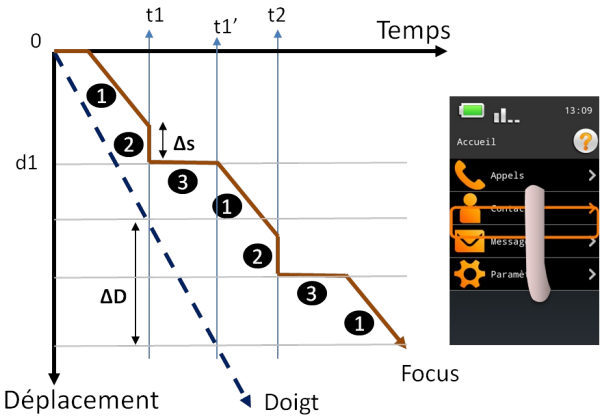

Figure 4: loi de déplacement du focus

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. Pour la navigation en largeur, la loi de déplacement du focus « en trois phases » assure déjà un bon couplage perception/action, rendant la manipulation intuitive. Il n'en va pas de même pour la navigation en profondeur dans la hiérarchie du menu (commandes *In* et *Out*) qui nécessite d'introduire un retour visuel approprié. Après avoir testé différents types d'effets visuels, nous avons opté pour un effet de décalage horizontal, vers la droite ou vers la gauche, portant sur l'élément focusé, ceci de façon corrélée avec le mouvement du doigt. La figure 5 montre cet effet de décalage dans le cas d'un glissé vers la gauche, et la figure 7 dans le cas d'un glissé vers la droite. Ce couplage fort entre le focus et le doigt a donc pour but de faire 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 en cours d'exécution (validation ou retour) liée à l'item sélectionné.

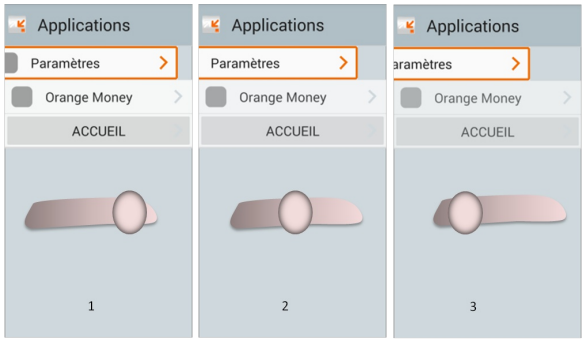

Figure 5 : effet visuel sur une commande de retour (trace du doigt en rose pâle)

De plus, ce *feedback* visuel progressif lui permet 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 lorsque le doigt reste en contact avec l'écran. Les deux effets visuels vus précédemment, chacun associé à un type de navigation et une direction (verticale ou horizontale), se combinent naturellement lors d'un geste composé, comme illustré par les figures 6 et 7.

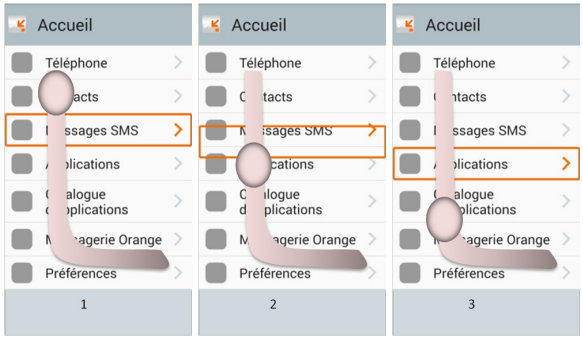

Figure 6 : effet visuel sur un geste composé lors de la phase de sélection

Enfin, nous avons mentionné plus haut que la validation d'un élément du menu pouvait être réalisée de deux façons : par un glissé vers la droite, sans contrainte de positionnement spatial, ou bien par un appui prolongé effectué directement sur l'élément cible. Dans ce dernier cas, l'effet visuel mis en œuvre, associé aux deux phases de sélection et de validation, se décompose comme suit : presque immédiatement après le contact du doigt sur l'écran la cellule de l'item pointé commence à se remplir d'une couleur, puis le focus de sélection saute sur l'item, enfin le remplissage s'étend à toute la cellule, annonçant la validation imminente.

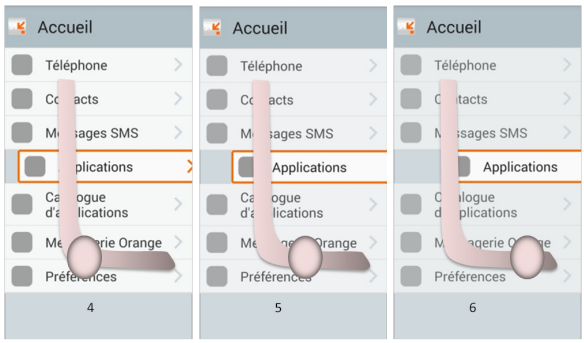

Figure 7 : effet visuel sur un geste composé lors de la phase de validation

Interaction gestuelle fine

La mise en œuvre de l'interaction gestuelle tactile au sein de la technique MenuDfA repose sur le framework d'interaction et de reconnaissance de gestes DGIL [13]. Ce dernier 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[1]. Dans cette perspective, le framework DGIL offre une approche unifiée du geste basée sur un langage d'interaction constitué de 36 primitives gestuelles. Ces primitives sont extraites à partir de l'analyse temps réel du geste. Elles sont autant d'événements significatifs que l'on peut connecter aux commandes et *feedbacks* de l'interface. Un geste est donc décrit par une séquence de primitives accompagnée de paramètres. Dans le cas d'un geste composé, comme celui des figures 6 et 7, la séquence produite est de la forme suivante :

`PRESS → START_MOVE → 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 → 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). La commande *Down* est déclenchée par la primitive `SEQ_MOVE_DOWN`. L'effet de

[1] Natural User Interfaces.

décalage horizontal à droite 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 la validation effective de la sélection. Précisons que le framework DGIL est ici configuré dans un mode spécifique dédié à la navigation séquentielle par positionnement relatif.

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), sachant que la partie reconnaissance de gestes est prise en charge par le framework DGIL. Mais afin d'obtenir une plus grande flexibilité dans le design de l'interaction, un *micro-framework*, nommé AEvent, a été développé afin de faciliter d'une part, la gestion événementielle, d'autre part, la sauvegarde en base de données des paramètres de préférences utilisateur. AEvent permet notamment 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, ceci 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. Grâce à AEvent il est également possible de définir des profils utilisateurs, voire de personnaliser l'interaction.

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 ou bien dynamiquement. 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). 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 navigation au moyen d'un focus de sélection, 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.

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[2]. 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. Trois besoins de manipulation ont été identifiés. L'interface proposait en conséquence trois profils d'interaction : le premier basé sur des «tapés» du dos de la main, permettant de progresser séquentiellement (tapé court) et de valider l'item (tapé long), le second basé sur des «glissés» (sélection) et «tapés long» (validation), et le troisième sur un défilement automatique du focus (sélection) associé à un «tapé» court ou long (validation). Ce prototype nous a permis de tester les principes de la navigation hiérarchique dans le cas d'une utilisation motrice contrainte. Enfin, la version actuelle du composant MenuDfA est le résultat d'un processus itératif de conception sur plusieurs mois, au niveau technique et ergonomique, alimenté par de nombreux retours d'usage issus de personnes standards, notamment novices, de personnes en situations de contraintes motrices et/ou visuelles.

CONCLUSION

Nous avons présenté dans cet article une nouvelle technique de navigation au doigt dans une interface tactile de type menus ou listes hiérarchiques. Dans cette approche l'utilisateur ne manipule qu'un seul objet, un focus de sélection, dans une logique séquentielle et avec des « glissés » continus, verticaux ou horizontaux. Elle implique une séparation entre les deux espaces, visuel et moteur, et s'applique de façon homogène dans toute la hiérarchie des menus. Des retours visuels et tactiles innovants sont mis en œuvre afin d'optimiser le contrôle utilisateur. Ces diverses propriétés, leur caractère flexible, et les premiers tests que nous avons pu réaliser, confirment que la technique MenuDfa est une réponse appropriée à la navigation tactile dans le contexte d'une

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

conception pour tous. De part ses caractéristiques, elle est bien adaptée aux terminaux mobiles de petite taille où les contraintes de précision gestuelle sont fortes. Des évaluations expérimentales plus poussées sont en cours afin de valider point par point les paradigmes proposés.

Enfin, compte tenu du nombre important de paramètres d'interaction impliqués (visuels, gestuels, tactiles, audio...), se pose la question de recherche de la meilleure façon de gérer cette complexité à la fois du point de vue du système et de l'utilisateur, afin d'aller vers le multiprofil et une personnalisation de l'interaction.

REMERCIEMENTS

Les auteurs remercient Stéphane Coutant pour la réalisation des premiers prototypes, Christophe Maldivi pour les évolutions de DGIL, Sophie Zijp-Rouzier pour sa contribution à la conception ergonomique, et Magalie Gimenes pour l'architecture et le développement logiciel de la version actuelle.

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).