Manipulation directe et accès indirect

Jonathan Aceituno

Manipulation directe et accès indirect

Jonathan Aceituno
Inria Lille – Nord Europe
IRCICA, 50 avenue Halley
59658 Villeneuve d'Ascq, France
jonathan.aceituno@inria.fr

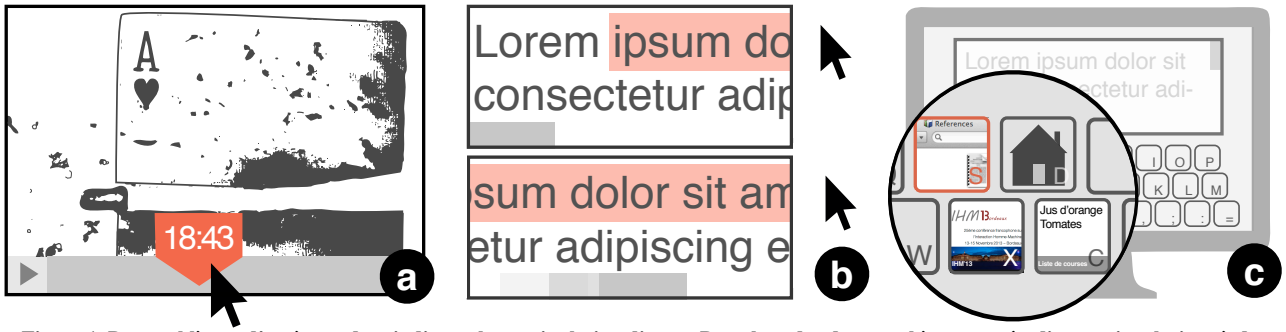

Figure 1. Des problèmes d'accès rendent indirecte la manipulation directe. Dans le cadre de cette thèse, nous étudions trois solutions à des problèmes de ce type : (a) augmenter la précision de la manipulation avec l'interaction subpixel, (b) augmenter sa portée avec le défilement implicite périphérique, et (c) faciliter l'accès à des objets distants avec Hotkey Palette.

RÉSUMÉ

La manipulation directe est un principe de conception souvent utilisé dans les interfaces graphiques. L'aspect direct de l'interaction à l'origine de ses avantages est cependant remis en question dans de nombreuses situations où l'accès aux objets d'intérêt est indirect : lorsque l'espace de manipulation n'est pas assez détaillé, lorsqu'il ne peut pas être affiché en entier, ou bien lorsque l'objet recherché réside dans un autre espace. Dans cet article, nous précisons le cadre théorique de la problématique de la thèse en cours et décrivons l'état de notre travail à travers trois solutions à ces problèmes d'accès indirect de la manipulation directe : l'interaction subpixel pour augmenter la précision de la manipulation, le défilement implicite périphérique pour augmenter sa portée, et l'assignation dynamique et contextuelle de raccourcis-clavier pour accéder rapidement à des objets.

Mots Clés

manipulation directe, interaction subpixel, résolution utile, défilement implicite périphérique, raccourcis clavier

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User interfaces – Graphical User Interfaces (GUI).

INTRODUCTION

De nombreux systèmes interactifs reposent sur un environnement graphique dont les objets peuvent être manipulés par l'utilisateur d'une manière similaire à ce qu'il ferait dans la réalité. Ces systèmes relèvent de la manipulation directe [8] en ce qu'ils possèdent les caractéristiques suivantes : l'objet d'intérêt est continuellement représenté ; l'utilisation d'une syntaxe complexe est remplacée par des actions physiques ; les opérations sont rapides, incrémentales et réversibles, et leur impact sur l'objet d'intérêt est visible immédiatement.

Le caractère direct de tels systèmes est appréciable car il implique un effort cognitif réduit pour l'utilisateur. Cependant, lorsque l'objet d'intérêt n'est pas immédiatement accessible, l'interaction conserve-t-elle ces propriétés ? Se rendre à une image bien précise d'un film d'1h30 dans un lecteur vidéo en utilisant une glissière de 200 pixels de long est impossible : pour un déplacement d'un pixel de la glissière, on saute de 27 secondes dans le film. Sélectionner un long texte lorsqu'on voit le début mais pas la fin impose de faire défiler soi-même la fenêtre avant de pouvoir effectivement faire la sélection. Retrouver rapidement un objet qui sert à la tâche courante mais qui n'est pas sous nos yeux demande de naviguer dans un ou plusieurs ensembles disjoints : le système de fichiers, le web, ou les fenêtres ouvertes. Ces exemples suggèrent que la manipulation directe ne passe pas à l'échelle à cause de certains problèmes d'accès qui la rendent indirecte.

Après une présentation détaillée de la problématique de cette thèse, le reste de cet article détaillera l'avancement de nos recherches à travers trois propositions pour résoudre les problèmes d'accès indirect (Figure 1) : l'interaction subpixel pour augmenter la précision de la manipulation directe, le défilement implicite périphérique pour augmenter sa portée, et l'assignation dynamique et contextuelle de raccourcis-clavier pour l'accès rapide à des objets.

MANIPULATION DIRECTE, ACCÈS INDIRECT

Dans cette section, nous présentons les différents constituants de la manipulation directe, dont nous nous servons ensuite pour détailler des problèmes d'accès ayant un impact négatif sur le caractère direct de l'interaction.

Manipulation directe

Deux concepts cohabitent dans l'idée de manipulation directe : la *distance cognitive* (*i.e.* entre l'interface et les intentions de l'utilisateur) et l'*engagement* (à quel point l'utilisateur se sent impliqué dans l'action) [5]. La distance cognitive s'exprime en deux dimensions : *distance sémantique* (à quel point les concepts utilisés dans l'interface expriment de manière concise les intentions de l'utilisateur), et *distance articulatoire* (à quel point les concepts utilisés dans l'interface correspondent à leur manifestation physique). Le degré d'engagement est faible pour une interface conversationnelle, alors qu'un monde virtuel, où l'action sur les objets d'intérêt est possible sans intermédiaire, implique un fort degré d'engagement. L'impression de directitude (*directness*) donnée par la manipulation directe est le résultat d'une distance cognitive réduite et d'un degré d'engagement maximal.

Accès indirect

Frohlich a proposé un ensemble de tâches pour lesquelles la manipulation directe n'est pas adaptée [4]. Parmi elles, deux nous intéressent particulièrement : utiliser des objets qui ne sont pas immédiatement visibles, et spécifier des actions avec une grande précision. Ces tâches suggèrent une problématique de l'accès aux objets d'intérêt dans le cadre de la manipulation directe : lorsque les objets d'intérêt ne sont pas immédiatement accessibles, des opérations d'accès supplémentaires sont nécessaires et réduisent alors la directitude des manipulations.

Un objet peut être inaccessible soit parce qu'il n'est pas dans une partie visible de l'espace de manipulation courant, soit parce qu'il réside dans un autre espace de manipulation. Il est alors nécessaire de faire précéder la manipulation d'une phase de navigation spatiale (*pan* ou *zoom*) dans le premier cas, ou d'accès ponctuels (par exemple, retrouver un fichier dans un système de fichiers hiérarchique) dans le second cas. Cette phase préliminaire augmente la distance sémantique de la manipulation : plus de concepts sont nécessaires pour exprimer la même intention. Elle augmente également la distance articulatoire : l'action physique effectuée ressemble moins à ce qui est exprimé dans le système. Ainsi, la manipulation directe l'est beaucoup moins lorsque l'objet d'intérêt n'est pas immédiatement accessible.

Dans le cadre de cette thèse, nous recherchons des solutions à ces problèmes d'accès qui minimisent les distances sémantiques et articulatoires. Plus spécifiquement, nous nous intéressons aux trois situations suivantes :

1. la vue sur l'espace de manipulation n'est pas assez détaillée pour permettre d'accéder à l'objet d'intérêt (navigation spatiale) ;
2. l'objet d'intérêt réside dans l'espace de manipulation actuel, mais est trop éloigné de la zone actuellement visible (navigation spatiale) ;
3. l'objet d'intérêt réside dans un autre espace auquel l'utilisateur a souvent accès (accès ponctuels).

Dans les sections suivantes, nous détaillons l'état de nos recherches concernant chacune de ces situations.

AUGMENTER LA PRÉCISION DE LA MANIPULATION

Imaginons qu'Hubert souhaite réaliser le plan du château fort que son association doit restaurer en utilisant un logiciel de conception assistée par ordinateur (CAO). Après avoir dessiné les murs du donjon, il doit l'orner de plusieurs drapeaux dont les positions doivent être scrupuleusement respectées pour des raisons historiques. Au bout de plusieurs essais, Hubert se rend compte qu'il ne peut pas les positionner correctement en les déplaçant directement, car le plus petit déplacement possible (un pixel) correspond à un mètre à l'échelle du plan. D'autres situations illustrant ce problème de précision peuvent être trouvées, par exemple, dans des tâches de positionnement dans une vidéo, d'édition d'images vectorielles, ou de navigation avec des lentilles focus+contexte [3].

Les solutions existantes à ce problème font appel à une spécification explicite du niveau de zoom sur l'espace en question (par exemple, en réponse à l'appui sur une touche). Mais étant donné que l'utilisateur doit penser à intégrer la gestion du niveau de zoom dans la planification de son action, ces solutions réduisent la directitude de la manipulation. Dans le monde réel, nous ne spécifions pas de zoom pour agir précisément : la précision est contrôlée au niveau sensori-moteur. Suivant cette observation, nous avons proposé l'*interaction subpixel* [7] (Figure 1a), une manière d'augmenter la précision de la manipulation sans réduire sa directitude, en tirant parti de la précision du mouvement humain qu'il est possible de capturer grâce aux périphériques de pointage haute-résolution actuels [1].

Interaction subpixel

L'idée sous-jacente est la suivante : si le pixel est la limite de ce qui peut être affiché, il ne doit pas limiter ce qui peut être manipulé. Pendant la manipulation dans le domaine subpixel, l'absence de retour visuel à l'emplacement de l'objet manipulé doit être compensé par un retour secondaire. Notons que ce retour existe déjà dans beaucoup de cas : il s'agirait de coordonnées précises ou de lentilles dans le cas d'un logiciel de CAO, ou bien de l'image actuelle du film et de l'horodatage dans le cas d'un lecteur vidéo.

Les systèmes d'exploitation des ordinateurs personnels gèrent le pointage indirect, où les déplacements captés par le périphérique de pointage indépendant de l'écran sont convertis en un déplacement du pointeur affiché à l'écran par l'intermédiaire d'une fonction de transfert. Nous avons identifié deux problèmes d'implémentation qui rendent impossible l'interaction subpixel dans les systèmes actuels : d'une part, la fonction de transfert ne tient pas compte de la précision du périphérique de pointage, et si c'était le cas, cette précision serait perdue car, d'autre part, la position du pointeur est rapportée en nombres entiers.

Une implémentation du système de pointage qui ne comporte pas ces problèmes permet alors l'interaction subpixel. Cependant, la quantité de précision gagnée n'est pas suffisante pour certaines applications. Nous avons donc proposé une méthode [7] pour assurer la manipulation d'un

1. La résolution des souris actuelles va de 400 à 10 000 *counts* par pouce (CPI), ce qui est élevé en comparaison avec la densité de pixel moyenne d'un écran, 150 pixels par pouce (PPI).

objet au niveau de précision voulu, en adaptant dynamiquement la fonction de transfert de manière à générer un déplacement optimal du pointeur (*i.e.* correspondant à la précision voulue) pour les plus petits déplacements captés par le périphérique, sans affecter le comportement de la fonction de transfert dans le domaine visible. Grâce à ces ajustements, on peut donc augmenter la précision de la manipulation directe de manière directe, en s'appuyant à la fois sur la précision du périphérique de pointage et de la personne qui le manipule.

Résolution utile

Si les limites des périphériques de pointage sont connues (lorsqu'elles sont documentées par le fabricant), celles de l'utilisateur restent à définir. Le niveau de précision désiré pour la manipulation d'un objet (par exemple, une seconde de vidéo) correspond à un déplacement qui est détecté par le périphérique de pointage dans l'espace moteur, et que l'utilisateur doit être capable de réaliser. Afin de le savoir, nous avons proposé la notion de *résolution utile* de l'utilisateur, c'est-à-dire la distance du plus petit déplacement qu'il peut produire de manière fiable avec le périphérique. Nous avons donc défini un protocole expérimental pour déterminer cette distance, et nous avons mesuré une résolution utile autour de 300 CPI (0,085 mm) pour une souris de 6400 CPI [1].

Ce travail nous a permis de déterminer que l'adaptation de la fonction de transfert jouera un rôle important dans l'interaction subpixel. Cependant, ce n'est pas encore suffisant pour déterminer quel degré de précision est accessible en pratique. La nature du retour visuel utilisé pendant l'interaction subpixel a de grandes chances d'influencer les performances de l'utilisateur. L'adaptation de la fonction de transfert peut également poser problème : plus la résolution utile est proche de la distance requise pour générer des déplacements de pointeur d'un pixel, plus la transition de la fonction de transfert entre le domaine subpixel et le domaine visible est abrupte et peut avoir des effets indésirables.

AUGMENTER LA PORTÉE DE LA MANIPULATION

Supposons qu'Hubert a fini le donjon et les autres tours internes du château. Il doit maintenant dessiner les remparts extérieurs grâce à l'outil *Mur* de son logiciel de CAO. Au bout de plusieurs essais, Hubert se rend compte qu'il ne peut pas dessiner un des murs en une fois : ce mur étant très étendu, Hubert ne peut voir à la fois son début et sa fin sans réduire fortement le niveau d'échelle actuel. D'autres tâches peuvent également présenter ce problème de portée : par exemple, la sélection de texte, la duplication de cellules dans un tableur, ou la manipulation de fichiers dans de grands dossiers.

Le problème ne peut pas être évité pendant la manipulation de l'objet, car cette dernière nécessite d'être maintenue (par exemple, en laissant le bouton gauche de la souris appuyé) d'une manière quasi-modale [6] qui exclut l'interaction avec d'autres instruments de navigation (par exemple, une barre de défilement). L'utilisation d'un canal d'entrée parallèle (par exemple, la molette de défilement de la souris) peut être envisagée, mais la coordination motrice nécessaire peut rendre cette action difficile, et beaucoup d'applications ne la supportent pas bien. Hubert doit alors décomposer la tâche en plusieurs phases alternées de navigation et de manipulation. Une approche plus directe peut être trouvée dans les systèmes actuels : pendant la manipulation, lorsque le pointeur dépasse la vue contenant l'objet d'intérêt, celle-ci commence à défiler à une certaine vitesse (Figure 1b). Cette technique est connue dans plusieurs boîtes à outils de création d'interfaces graphiques en tant qu'*autoscroll*, mais nous préférons le terme moins ambigu de *défilement implicite périphérique*.

Défilement implicite périphérique

Cette technique pourtant commune n'a encore fait l'objet d'aucun travail de recherche. Les implémentations existantes ont des comportements très différents et il n'est pas rare qu'un utilisateur évite de les utiliser parce qu'il n'arrive pas à les contrôler. Ces observations nous ont conduit à étudier l'interaction avec cette technique de manière à dégager des heuristiques pour concevoir un défilement implicite périphérique qui minimise l'effort cognitif et maximise les performances. Le défilement implicite périphérique étant le plus souvent implémenté pour la sélection de texte, nous nous sommes concentrés sur cette tâche.

Nous avons d'abord développé des méthodes pour extraire les comportements de 11 systèmes actuels, dont quatre boîtes à outils de création d'interfaces graphiques (Cocoa, GTK, Java, Qt), deux moteurs de navigateurs web (Gecko, WebKit), et cinq applications (Adobe Photoshop, Adobe Reader, Microsoft Word 2010, le Bloc-notes et l'Explorateur de Windows). Pour examiner chaque comportement, nous instrumentons une vue défilable qui l'implémente en synthétisant des séquences d'événements de souris pour simuler une manipulation en dehors de la vue, et en enregistrant tous ses défilements. Selon l'implémentation, nous avons utilisé différentes méthodes pour récupérer les informations de défilement : écriture d'un script, utilisation des interfaces d'accessibilité du système, ou analyse de séquence vidéo par *template matching*. Notre analyse a révélé que la majorité des comportements (comme celui de WebKit) utilisaient un défilement automatique avec des vitesses variées contrôlées par la distance entre le pointeur et le bord de la vue, tandis que d'autres défilaient uniquement lorsque le pointeur était déplacé (comme celui de GTK).

Nous avons ensuite comparé six comportements représentatifs de la diversité rencontrée. Nous avons mesuré la performance (temps de sélection) et la charge de travail perçue sur 18 participants pour une tâche de sélection de texte où les six méthodes de défilement implicite périphérique devaient être utilisées pour quatre différentes longueurs de texte à sélectionner. Cette expérience contrôlée a permis de déterminer que les comportements automatiques proches de celui de WebKit donnent la meilleure performance, la plus faible charge de travail perçue, et sont préférés par la majorité des participants. Les comportements manuels proches de celui de GTK donnent une performance respectable, mais une plus forte charge de travail perçue, et très peu de participants les préfèrent. Une analyse plus détaillée sera disponible ultérieurement dans un article en cours de rédaction.

FACILITER LA MANIPULATION D'OBJETS DISTANTS

Imaginons maintenant que le plan de château fort d'Hubert est presque terminé et qu'il lui reste à réaliser le pont-levis et les douves. Son association a déjà travaillé sur la restauration de ce pont-levis et il en a un plan dont il peut reprendre certaines parties. Il se souvient également qu'il possède quelques photos qui lui permettront de récupérer la couleur si particulière de l'eau des douves. Hubert se rend compte que ces documents ne sont pas accessibles sur son écran, et qu'il doit naviguer dans son système de fichiers afin de les retrouver.

Certaines interfaces, comme le bureau ou le Dock d'OS X, offrent un accès direct et rapide à des ressources qui ne résident pas dans l'espace de manipulation courant. Cependant, leur capacité limitée fait que l'utilisateur ne peut pas souvent trouver ce qu'il cherche. Il doit alors se rabattre sur la navigation dans une hiérarchie de fichiers. Les utilisateurs organisant leurs ressources selon leur contexte d'utilisation [9], les interfaces d'accès rapide devraient offrir des mécanismes de contextualisation ("quand je suis *ici*, je veux avoir accès à *ça*") pour être plus utiles. Des techniques actuelles le permettent, mais elles sont spécifiques à la gestion des fenêtres (par exemple, les bureaux virtuels). Nous avons donc conçu *Hotkey Palette* [2], une technique d'accès rapide à des ressources s'appuyant sur des raccourcis-clavier configurables et contextuels.

Hotkey Palette

Hotkey Palette se présente sous la forme d'un clavier virtuel (Figure 1c) qu'il est possible d'afficher en maintenant une touche du clavier appuyée (*Fn*, dans notre implémentation). L'appui sur *Fn+X* permet d'associer la fenêtre courante, ainsi que le document qu'il représente, à la touche X. Il est ensuite possible d'accéder au document en appuyant sur *Fn+X* depuis n'importe où. On peut également aller le chercher avec la souris en laissant la touche *Fn* appuyée. Un document peut aussi être associé à une touche en le glissant sur cette dernière. La symétrie entre le clavier virtuel et le clavier physique assure la directitude de l'interface.

Hotkey Palette permet une contextualisation flexible des raccourcis-clavier s'appuyant sur la hiérarchie des ressources. Pour une touche, il est possible d'associer un raccourci-clavier à une ressource dans un premier contexte, et à une autre ressource dans un second contexte. Le contexte actuel, spécifié par la fenêtre courante, détermine alors à quelles ressources il est possible d'accéder. En associant les fenêtres ouvertes aux documents qu'ils représentent, Hotkey Palette permet de localiser chaque ressource (à l'exception des fenêtres de configuration, par exemple) dans une hiérarchie. Cette dernière est prise en compte dans la contextualisation : une fenêtre associée à *Fn+S* dans le contexte du répertoire `Travail` pourra être retrouvée dans le contexte du document `Château-fort` si ce dernier se trouve quelque part dans le répertoire `Travail`.

CONCLUSION

La problématique de cette thèse est l'aspect indirect de l'accès à des objets d'intérêt dans le contexte de la manipulation directe. À travers l'étude de ce contexte, nous avons isolé trois aspects du problème : le manque de précision, le manque de portée, et la difficulté d'accéder à des objets distants. Pour chacun de ces aspects, nous avons montré le caractère indirect des solutions existantes, puis étudié de nouvelles techniques d'interaction plus directes : l'interaction subpixel permet d'augmenter la précision de la manipulation, le défilement implicite périphérique permet d'augmenter la portée de la manipulation, et Hotkey Palette permet de faciliter l'accès à des objets distants.

Nous envisageons de continuer ces travaux en suivant plusieurs axes. Tout d'abord, le contrôle implicite du défilement pendant la manipulation directe peut être ambigu lors du déplacement d'un objet d'une vue à l'autre. Nous sommes en train de mettre au point une expérience contrôlée pour comparer différentes techniques permettant de lever cette ambiguïté. Ensuite, nous souhaiterions caractériser plus formellement les avantages offerts par la palette de raccourcis clavier et clarifier la contribution de la contextualisation des raccourcis. Enfin, certaines limites de l'interaction subpixel restent à définir au niveau du retour visuel et de l'adapation de la fonction de transfert.

BIBLIOGRAPHIE

1. Aceituno, J., Casiez, G., and Roussel, N. How low can you go ? : human limits in small unidirectional mouse movements. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 1383–1386. `http://www.youtube.com/watch?v=59R-CxMsKoA`.
2. Aceituno, J., and Roussel, N. The Hotkey Palette : Flexible Contextual Retrieval of Chosen Documents and Windows. Rapport de recherche RR-8313, Inria, June 2013.
3. Appert, C., Chapuis, O., and Pietriga, E. High-precision magnification lenses. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, ACM (New York, NY, USA, 2010), 273–282.
4. Frohlich, D. M. The history and future of direct manipulation. *Behaviour & Information Technology 12*, 6 (1993), 315–329.
5. Hutchins, E. L., Hollan, J. D., and Norman, D. A. Direct manipulation interfaces. *Hum.-Comput. Interact. 1*, 4 (Dec. 1985), 311–338.
6. Raskin, J. *The humane interface : new directions for designing interactive systems*. Addison-Wesley, Mar. 2000.
7. Roussel, N., Casiez, G., Aceituno, J., and Vogel, D. Giving a hand to the eyes : leveraging input accuracy for subpixel interaction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, ACM (New York, NY, USA, 2012), 351–358. `http://www.youtube.com/watch?v=Cwxyksa7iho`.
8. Shneiderman, B. Direct manipulation : A step beyond programming languages. *Computer 16*, 8 (1983), 57–69.
9. Zacchi, A., and Shipman, F. Personal environment management. In *Research and Advanced Technology for Digital Libraries*, L. Kovács, N. Fuhr, and C. Meghini, Eds., vol. 4675 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, 345–356.