Pointage Sémantique et Distracteurs, la Dynamique du Pointage à la Rescousse

Renaud Blanch

LRI & INRIA Futurs
Université Paris-Sud XI
91 405, Orsay CEDEX, France
renaud.blanch@lri.fr

RESUME
Le pointage sémantique permet de faciliter le pointage en attribuant indépendamment des tailles visuelle et effective différentes à tous les objets graphiques d'une interface. Ces tailles sont choisies en fonction d'une information jusqu'à maintenant négligée par le système : l'importance pour l'interaction de chaque zone de l'écran. Nous améliorons cette technique en prenant en compte la trajectoire du curseur, qui comporte aussi des informations sous-exploitées, pour atténuer l'impact négatif des cibles potentielles présentes sur la trajectoire d'un pointage : les distracteurs.

MOTS CLES : distracteur, pointage, pointage sémantique, ratio *control/display*.

INTRODUCTION

La tâche de pointage est l'interaction élémentaire des interfaces graphiques. L'augmentation de la complexité ou, au mieux, du nombre des fonctionnalités des logiciels, se traduit par un nombre sans cesse croissant de cibles potentielles alors que la taille des écrans n'évolue guère [2]. On assiste donc à une prolifération de cibles de plus en plus nombreuses et de plus en plus petites. La loi de Fitts [6] nous rappelle que plus une cible est loin du curseur et petite, plus le temps pour l'atteindre est long.

C'est dans ce contexte que des travaux récents ont tenté, par différentes approches, de faciliter les tâches de pointage. Nous donnons ici un aperçu de ces techniques, et en particulier du pointage sémantique que nous avons développé dans un travail antérieur [3]. Nous montrons ensuite que cette technique peut être enrichie en utilisant des informations extraites de la trajectoire du curseur, ce qui la rend encore plus efficace.

TRAVAUX ANTERIEURS

La loi de Fitts [6] nous enseigne que le temps pour atteindre une cible croît avec la distance *D* à cette cible et diminue lorsque sa taille *W*, elle, augmente. Partant de ce simple constat, de nombreuses méthodes ayant pour objectif de faciliter le pointage ont été proposées. En voici quelques unes.

Faciliter le Pointage en Réduisant *D*

Réduire la distance des cibles est sans doute la première idée qu'on peut avoir. Cette idée a donné le jour aux menus contextuels qui apparaissent au plus près du curseur, réduisant ainsi la distance des entrées du menu au curseur au minimum. Les *pie menus* de Hopkins [4] améliorent encore cette technique en disposant les éléments du menus en cercle autour du curseur, les mettant ainsi à une distance quasiment nulle de celui-ci (figure 1).

Figure 1 *:* Les entrées d'un *Pie menu* disposées en cercle autour du curseur.

Le *drag-and-pop* de Baudisch [1] est une technique qui consiste à rapprocher les cibles potentielles d'un *drag-and-drop* vers l'objet déplacé. Ces cibles potentielles sont déterminées par la direction du déplacement et la compatibilité des objets (figure 2).

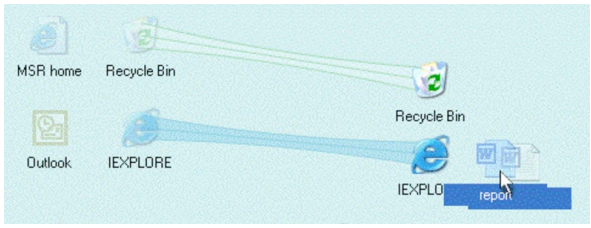

Figure 2 : Un document est déplacé et le *drag-and-pop* approche les cibles susceptibles de lui convenir.

Enfin, le pointage d'objets [7] est une technique récente qui pousse ce genre d'idée à l'extrême en "supprimant" artificiellement l'espace vide entre les objets du bureau, réduisant par là-même *D* le plus possible. Cette suppression artificielle est obtenue en faisant sauter le curseur vers l'objet le plus proche dans la direction de son mouvement dès qu'il se retrouve dans le vide.

Faciliter le Pointage en Augmentant *W*

L'autre variable sur laquelle il est possible de jouer est la taille de la cible. Kabbash et Buxton montrent qu'en utilisant pour la sélection une zone *(area cursor)* plutôt qu'un point (pointe du curseur), cela revient pour les cibles qui sont plus petites que la zone de sélection choisie, à leur donner une taille égale à cette zone [8]. Cette technique permet donc de faciliter la sélection de petites cibles en leur donnant artificiellement une taille supérieure, mais pose des problèmes lorsque plusieurs cibles entrent simultanément dans la zone de sélection.

Une autre approche consiste à dilater les cibles lorsque le curseur s'en rapproche par un effet de *fish-eye* ou en leur adjoignant des poignées ou *"bubble"* [9, 5]. Une telle technique est utilisée dans le Dock de Mac OS X (figure 3).

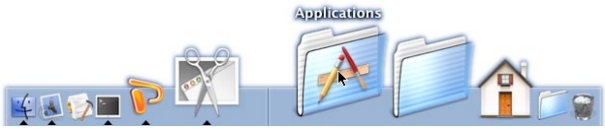

Figure 3 : Les cibles du Dock de Mac OS X se dilatent à l'approche du curseur.

Des comparaisons avec d'autres techniques montrent que la dilatation des cibles est efficace même si elle est mise en jeu tard dans le mouvement, comme l'ont montré McGuffin et Balakrishnan [9], et que l'on en tire parti même si elle n'est pas anticipée, comme l'ont montré Zhai et al. [10]. Cependant, ces techniques posent aussi problème lorsque plusieurs cibles sont proches. Il faut alors en effet les déplacer pour faire de la place à la cible dilatée. Dans le cas du Dock, par exemple, Zhai et Beaudouin-Lafon ont montré que si l'expansion permet de mieux repérer les cibles, elle n'offre par contre aucune aide au pointage puisque le mouvement des cibles généré par le déplacement du centre de la dilatation annule le bénéfice de celle-ci [10].

Une autre voie explorée récemment ne s'attaque plus de front à *D* et à *W,* mais cherche à aboutir aux mêmes effets sans modifier le placement et la taille des objets à l'écran. Il s'agit de l'adaptation du ratio *control/display*.

L'ADAPTATION DU RATIO *CONTROL/DISPLAY*

L'idée est de faciliter le pointage en modifiant le ratio *control/display* au cours du mouvement. Le ratio *control/display* est la constante qui lie les déplacements du curseur à ceux de la souris. Le même déplacement du curseur est provoqué par un mouvement de la souris de plus en plus grande amplitude à mesure que le ratio augmente. En adaptant cette constante à divers paramètres au cours du pointage, on peut artificiellement modifier, pour le mouvement à accomplir avec la souris et non pour le curseur à l'écran, la taille et la distance des cibles.

L'"Accélération" de la Souris

En fait, une telle adaptation existe depuis longtemps dans ce qui est nommé improprement l'"accélération" de la souris. Cette technique change le ratio en fonction de la vitesse de la souris, faisant en sorte que lorsque la souris se déplace vite, la distance couverte par le curseur pour un déplacement donné de la souris soit plus grande que lorsque celle-ci se déplace lentement. Cela permet par exemple d'aller d'un bout à l'autre de l'écran sans avoir besoin de beaucoup de place sur la table, tout en conservant une précision élevée pour des mouvements de faible amplitude réalisés à basse vitesse.

Le Pointage Sémantique

Nous avons montré par ailleurs que le ratio *control/display* peut s'interpréter comme un facteur d'échelle entre l'espace moteur (la table sur laquelle évolue la souris) et l'espace visuel (l'écran où sont affichés les objets de l'interface) [3]. En adaptant le ratio en fonction, cette fois, de la position du curseur et non de sa vitesse, on peut alors modifier les tailles relatives des différents objets dans l'espace moteur tout en préservant leur disposition à l'écran.

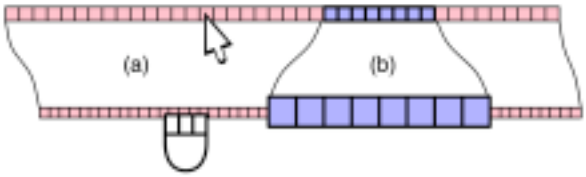

Figure 4 : Le ratio *control/display* comme facteur d'échelle entre l'écran (ligne de pixels en haut) et l'espace de la souris ("images" des pixels sur la table après un changement d'échelle en bas).

La figure 4 illustre cette idée mise en œuvre par le pointage sémantique. La ligne du haut représente une rangée de pixels à l'écran, et celle du bas, la position qu'ils occupent pour la souris, sur la table, après avoir subi un changement d'échelle. La zone (a) représente une partie non intéressante pour l'interaction (fond de l'écran par exemple). Elle est contractée pour être traversée rapidement. La zone (b) représente une cible importante. Elle est donc dilatée pour permettre à la souris de s'arrêter facilement dedans.

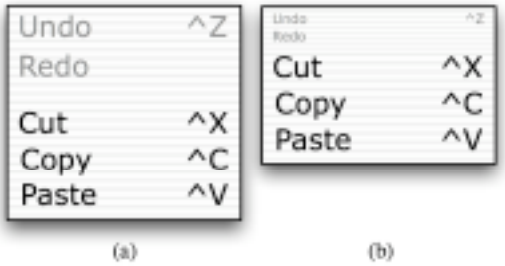

Figure 5 : Le menu redesigné.
(a) version à l'écran, et (b) version dans l'espace moteur qui permet de "sauter" les entrées inactives.

On peut ainsi contracter les zones de vide (ou celles qui ne sont simplement pas importantes pour l'interaction comme c'est le cas des entrées inactives du menu de la figure 5), on peut aussi réduire à l'écran des objets dont l'encombrement ne sert qu'à faciliter la manipulation, tout en préservant leur taille initiale pour une interaction aussi aisée (la figure 6 présente une telle évolution de la barre de défilement classique).

Figure 6 : La barre de défilement redesignée. (a) version originale, (b) version à l'écran qui réduit l'encombrement en présentant la même information, et (c) version dans l'espace moteur qui préserve l'utilisabilité du design original.

Pour mieux appréhender le pointage sémantique, on en trouvera à l'adresse suivante une démonstration interactive en ligne sous la forme d'une *applet* Java :
http://insitu.lri.fr/~blanch/project/SemanticPointing/demo/

Bénéfices du Pointage Sémantique

L'intérêt du pointage sémantique tient au fait que l'on peut, en choisissant pour un objet une taille à l'écran et un ratio, lui attribuer en fait deux tailles distinctes et indépendantes : sa taille dans l'espace visuel et sa taille dans l'espace moteur. Cette dichotomie permet de séparer les contraintes qui pèsent sur le choix de la taille des objets et de les faire peser sur l'un ou l'autre des aspect. La taille à l'écran peut ainsi n'être choisie qu'en fonction de l'information que doit présenter l'objet à l'utilisateur. Par contre, la taille dans l'espace moteur n'est régie que par la contrainte de l'utilisabilité, et on peut donc rendre gros dans cet espace les objets interactifs peu porteurs d'information comme les poignées de manipulation ou les boutons. L'exemple de la barre de défilement (figure 6) illustre ce principe : la barre donne peu d'information, on lui attribue peu de pixels (figure 6b). Par contre, les flèches aux extrémités doivent être facile à atteindre, elles sont donc grossies dans l'espace moteur (figure 6c).

Cette séparation en deux tailles n'a d'intérêt que parce que le temps de pointage donné par la loi de Fitts pour un objet est déterminé par sa taille et sa distance, mais celles-ci, comme nous l'avons montré par une expérimentation contrôlée dans [3], sont comprises dans l'espace *moteur*. On peut donc, étant donnée une tâche spécifiée visuellement, la rendre plus facile à réaliser par simple manipulation du ratio *control/display,* et ceci sans modifier le placement à l'écran des objets.

PRISE EN COMPTE DES DISTRACTEURS

Un problème inhérent à la plupart des techniques présentées jusqu'à présent, est que le système ne peut prédire la cible précise d'un mouvement. Il facilite également toutes les cibles considérées comme potentielles du mouvement. Les objets considérés comme cibles possibles par le système qui diffèrent de la cible réelle, celle que l'utilisateur a en tête, sont autant de distracteurs pour la réalisation de la tâche. Il s'agit, dans le cas d'un *pie menu* par exemple, de toutes les entrées du menu exceptée celle qui sera choisie. Dans le cas du pointage d'objet, comme dans celui du pointage sémantique, il s'agit des objets qui sont croisés par la trajectoire du curseur, et qui le ralentissent au passage. Dans le pointage normal, il s'agit de chaque pixel qui sépare le curseur de la cible puisque tous les pixels sont traités de la même manière par le système.

Un Ratio *Control/Display* Optimisé

Le système de pointage idéal ressemble à l'interface souvent présente dans les films de science-fiction : le bouton unique qui effectue précisément l'action que l'utilisateur a en tête. Si le système ne peut prédire la cible que l'utilisateur veut atteindre, certaines informations peuvent être exploitées simplement et de manière complètement déterministe pour régler le compromis vitesse de déplacement / précision des mouvements du curseur que représente aussi le ratio *control/display,* et ce de manière à faciliter le pointage.

Prise en Compte de la Position. La position du curseur à l'écran donne une information complètement négligée par la plupart des systèmes : le curseur est-il dans une cible ou dans une zone inintéressante pour l'interaction ? Cela revient à considérer le pointage d'une cible comme la sélection d'un des pixel qui la compose parmi tous ceux de l'écran et non comme la simple sélection d'une cible parmi un ensemble de cibles. Les pointages sémantique et d'objets [3, 7] s'appliquent justement à réintroduire cette information pour optimiser le ratio *control/display*.

Prise en compte de la vitesse. La vitesse de déplacement du curseur fournit elle aussi une information intéressante facile à exploiter. En effet, lorsque le curseur se déplace rapidement, il y a peu de chance pour que l'utilisateur ait l'intention de s'arrêter. En fait, plus il déplace la souris rapidement, plus il recherche la vitesse, et moins il est préoccupé par la précision. A contrario, pour sélectionner précisément une cible, les déplacements se feront à faible vitesse. C'est exactement de cette manière qu'est adapté le ratio *control/display* par l'"accélération" de la souris sur la plupart des systèmes existants.

Ces deux optimisations (position et vitesse) ne permettent cependant pas de lever une ambiguïté résiduelle. En

effet, les données de position et de vitesse ne discriminent pas un début de mouvement de la fin d'un pointage. Dans les deux cas, le curseur est en général à proximité d'une cible (but du mouvement précédent ou de celui en cours), sa vitesse est faible (le curseur est en train de démarrer ou en train de s'arrêter). Pour déterminer dans quel contexte le mouvement se situe, il faut donc prendre en compte une variable qui ne présente pas de similitude entre le début et la fin d'un mouvement.

Prise en compte de l'accélération. L'accélération du curseur est positive lorsque le début d'un mouvement est entamé, alors qu'elle devient négative lorsque l'on freine pour s'arrêter dans une cible. C'est donc le candidat idéal qui permet de discriminer entre le début d'un mouvement et sa fin.

En résumé, la prise en compte de la position du curseur permet de pointer des cibles et non plus de simples pixels sur l'écran, et ce faisant, de faciliter arbitrairement le pointage (figure 7, à gauche, une trajectoire typique sans aide au pointage, à droite, l'effet du pointage sémantique qui réduit significativement le temps de pointage pour la même tâche).

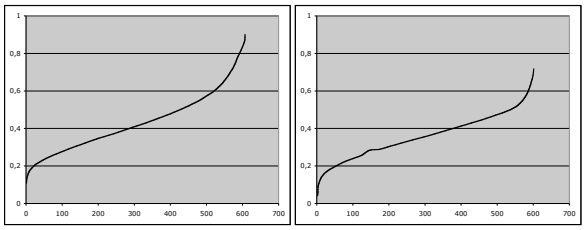

Figure 7 : Trajectoires typiques d'un pointage.
(abscisse : position du curseur, ordonnée : temps, cible en 600)
à gauche, sans le pointage sémantique,
à droite, avec la cible ayant une échelle de 4.

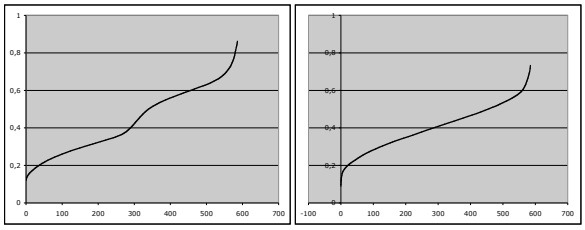

Figure 8 : Pointage sémantique avec un distracteur en 300.
à gauche, le pointage sémantique classique,
à droite, en tenant compte de la vitesse et de l'accélération.

Par contre, un problème se pose alors : celui des distracteurs qui freinent le curseur le long de sa trajectoire et déstabilisent l'utilisateur par un comportement non souhaité du curseur. Le bénéfice du pointage sémantique est alors minimisé (figure 8, à gauche on note la discontinuité dans la trajectoire causée par un distracteur à mi-chemin). En utilisant l'information de vitesse et d'accélération pour moduler le pointage sémantique, on arrive à atténuer l'effet des distracteurs et à revenir au gain initial sans distracteur (figure 8, à droite, le distracteur à mi-course n'a pas d'influence sur la trajectoire et on retrouve le gain observé sans le distracteur).

CONCLUSION

On a montré que l'utilisation d'informations issues de la trajectoire du curseur et de sa dynamique permettait d'aider le pointage. L'utilisation de la position a fait l'objet d'une étude statistique approfondie [3]. Celle de la vitesse et de l'accélération est prometteuse comme le montrent les données présentées ici (issue d'un prototype visant à valider l'idée, figure 7 et 8) mais elle mérite une étude contrôlée pour être pleinement validée.

BIBLIOGRAPHIE

1. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. Interact 2003,* pages 57–64, 2003.
2. M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proc. CHI 2000,* pages 446–443. ACM Press, 2000.
3. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. CHI 2004,* pages 519–526. ACM Press, 2004.
4. J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proc. CHI 1988,* pages 95–100. ACM Press, 1988.
5. A. Cockburn and A. Firth. Improving the acquisition of small targets. In *Proc. HCI 2003,* pages 181–196, 2003.
6. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psycology,* 47:381–391, 1954.
7. Y. Guiard, R. Blanch, and M. Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in GUIs. In *Proc. GI 2004,* 2004.
8. P. Kabbash, and W. Buxton. The "Prince" technique: Fitts'law and selection using area cursors. In *Proc. CHI 1995,* pages 273–279. ACM Press, 1995.
9. M. McGuffin and R. Balakrishnan. Acquisition of expanding targets. *In Proc. CHI 2002,* pages 57–64. ACM Press, 2002.
10. S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard. Human on-line response to target expansion. *In Proc. CHI 2003,* pages 177–184. ACM Press, 2003.