Programmer et Utiliser l'Interaction Gestuelle

Caroline Appert

Laboratoire de Recherche en Informatique
91405, Orsay, France
appert@lri.fr

RESUME
Cette présentation met en avant quelques barrières que rencontre l'interaction gestuelle à la fois de la perspective du développeur et de celle de l'utilisateur final. Elle propose quelques pistes à explorer pour lever ces barrières. Bien que les travaux mentionnés se placent dans le cadre de l'interaction avec un dispositif de pointage 2D (souris, stylet, doigt...), les principes des idées proposées méritent d'être explorées dans un contexte d'entrée 3D.

MOTS CLES : Interaction gestuelle, développement, découverte, apprentissage

INTRODUCTION
L'interaction gestuelle, que nous définissons ici comme l'entrée d'une suite de points continue pour déclencher une commande de l'application, est une des réponses à apporter aux limites dont souffre le paradigme de "pointage-puis-clic". Ceci est particulièrement vrai dans le cas d'une surface d'affichage de taille limitée qui présente des cibles petites et donc coûteuses à acquérir ou encore dans le cas des interfaces à entrée directe. En effet, dans ce dernier cas, le dispositif de pointage étant directement en contact avec la surface d'affichage, il est fréquent que certaines zones à désigner soient cachées. Avec une interaction gestuelle, il ne s'agit pas de désigner une zone spatiale parmi la surface d'affichage (cliquer sur l'élément de menu "couper") mais de tracer un symbole particulier parmi un vocabulaire (dessiner un 'x'), la surface d'affichage se trouvant ainsi "libérée" des cibles à acquérir. La modalité gestuelle semble donc particulièrement appropriée pour téléphones à écran tactile, tablettes écran ou encore tables interactives qui sont désormais très nombreux dans le commerce.

Cependant pour être largement disséminée dans nos interfaces et aisément adoptée par une large part des utilisateurs, l'interaction gestuelle doit être facile à programmer pour les développeurs d'interfaces graphiques et facile à appréhender pour les utilisateurs finaux. Ces deux conditions ne sont pas remplies à l'heure actuelle puisque les boîtes à outils de développement d'interfaces largement utilisées n'offrent *aucune facilité de développement pour la modalité gestuelle*. Ces interfaces sont donc coûteuses à développer et, lorsque l'effort de développement est fourni, elles présentent des problèmes d'utilisabilité pour l'utilisateur final. En effet, contrairement aux cibles graphiques visibles sur la surface d'affichage, *l'utilisateur ne voit pas les gestes disponibles* et se trouve donc confronté à de sérieux problèmes : la découverte et l'apprentissage. Cette présentation présente quelques travaux de recherche pour répondre à ses problèmes de développement et d'utiliisabilité.

PROGRAMMER L'INTERACTION GESTUELLE
Programmer l'invocation d'une commandes par un geste est coûteux car cela requiert entre autres de programmer l'algorithme capable de reconnaître un geste parmi un vocabulaire de gestes et de proposer un vocabulaire de gestes. Certaines boîtes à outils issues de la recherche comme SATIN [5] ou SwingStates [2] intègrent (i) un module de reconnaissance de gestes et (ii) une application pour associer un geste à un mot d'un vocabulaire. Elles pallient ainsi l'absence de la modalité gestuelle dans les boîtes à outils "grand public" comme Java Swing, Gtk ou Qt. Cependant le travail de conception et programmation ne se limite pas à ces deux tâches. Premièrement, l'imagination fait souvent défaut dans la phase d'association geste-commande. Ensuite, il faut programmer l'invocation effective de la bonne commande quand le geste correspondant est reconnu. Enfin, il faut implémenter des aides pour l'utilisateur final afin que celui-ci puisse découvrir les raccourcis gestuels disponibles.

Stroke Shortcuts Toolkit [3] va plus loin que les boîtes à outils sus-mentionnées et permet d'ajouter des raccourcis gestuels pour un coût de développement réduit au minimum. À partir du programme d'une interface Java Swing, le développeur n'a qu'une ligne de code à ajouter pour lancer un environnement avancé qui permet d'ajouter des raccourcis gestuels à l'interface. L'environnement propose alors la liste des commandes trouvées dans l'interface et un dictionnaire de gestes prédéfinis. Cet environnement visuel permet d'associer un geste à une commande très simplement et le développeur n'est pas confronté au "problème de la page blanche" grâce au dictionnaire de gestes prédéfinis. L'imagination est stimulée mais sans être contrainte car l'environnement permet aussi au développeur de définir lui-même un geste arbitraire qu'il juge plus approprié à une commande donnée.

Les associations geste-commande peuvent être ensuite chargées en une simple ligne de code dans l'interface Java Swing. Ainsi, sans aucun travail de programmation supplémentaire, un geste exécuté par l'utilisateur déclenche la bonne commande et l'interface se trouve augmentée pour permettre la découverte et l'apprentissage des raccourcis gestuels.

UTILISER L'INTERACTION GESTUELLE

Les aides à la découverte et à l'apprentissage pour l'utilisateur final disponibles dans SST sont :

- les *items de menu* augmentés : le raccourci gestuel est visuellement indiqué dans l'item de menu correspondant à la manière des raccourcis clavier,
- les *tooltips* augmentés : l'aide textuelle qui apparaît sous forme de tooltip lorsqu'un composant graphique est "survolé" indique le raccourci gestuel qui correspond à ce composant graphique,
- l'"antisèche" : lorsque l'utilisateur fait une pause au cours de l'exécution d'un geste, une fenêtre indiquant les associations geste-commande apparaît aussitôt.

Cependant ces aides ne sont pas optimales. Les deux premières qui *balisent les chemins connus* demande des actions supplémentaires et volontaires de la part de l'utilisateur : aller voir l'item de menu (auquel cas, il est aussi efficace de cliquer sur l'item directement, ce qui n'encourage pas l'apprentissage) ou survoler un composant jusqu'à l'affichage de son tooltip (ici encore, il est aussi efficace d'interagir directement avec le composant). La dernière, l'antisèche, est plus intéressante car elle apparaît "naturellement", c'est-à-dire en *réponse à une hésitation*. Elle est cependant extrêmement gourmande en termes de surface d'affichage puisqu'elle affiche la liste exhaustive des associations geste-commande. Récemment, Bau et Mackay ont proposé une aide plus élaborée, OctoPocus [4]. Le guide OctoPocus apparaît également en réponse à une hésitation mais il est contextuel, c'est-à-dire qu'il contient uniquement les gestes compatibles avec le début de geste déjà entré par l'utilisateur. Aussi, le contenu du guide est mis à jour dynamiquement au fur et à mesure de l'entrée de l'utilisateur. La figure 1 illustre le fonctionnement : non seulement le guide ne contient que les chemins des gestes compatibles avec le geste en cours mais aussi l'épaisseur d'un chemin est proportionnelle à la plausibilité du geste.

Ce type d'aide à la découverte et à l'apprentissage exige la capacité de reconnaître un geste incomplet. Appert et Bau [1] ont proposé un algorithme qui permet non seulement cette reconnaissance d'entrée partielle mais aussi une estimation de l'échelle du geste incomplet pour maintenir un guide visuellement cohérent quelle que soit l'entrée gestuelle. En effet, il est extrêmement fréquent que le même geste à différentes échelles correspondent à la même commande (une petite ou une grande boucle invoqueront la commande **cut** sur l'exemple de la figure 1). Cet algorithme de reconnaissance de geste incomplet

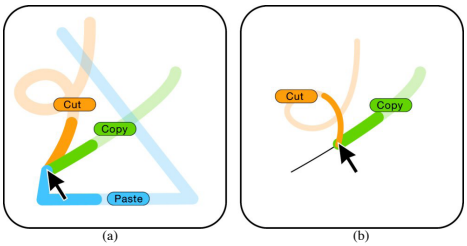

Figure 1 **:** (a) Le guide OctoPocus affiche trois chemins de gestes possibles. (b) L'exécution du geste **copy** entraîne la disparition de **paste** qui n'est plus compatible et la diminution du chemin de **cut** qui devient moins plausible.

n'est pas seulement utile pour OctoPocus en particulier mais pour toutes les techniques d'aide contextuelles en général. Par exemple, il peut être utilisé pour réduire la surface d'affichage occupée par l'antisèche puisqu'il ne s'agit plus ici de proposer la liste exhaustive de commandes mais uniquement celles qui sont pertinentes par rapport à un début de geste donné.

CONCLUSION

Cette présentation propose des pistes à explorer pour faciliter le développement d'interfaces supportant la modalité gestuelle et pour encourager son adoption par les utilisateurs finaux. Les pistes sont proposées dans le contexte d'applications graphiques en 2D et méritent encore d'être explorées dans un contexte plus "libre" comme l'exécution de gestes en 3D "dans l'air".

BIBLIOGRAPHIE

1. C. Appert and O. Bau. Scale detection for a priori gesture recognition. In *Proc. CHI '10*, xxx–xxx. ACM.
2. C. Appert and M. Beaudouin-Lafon. SwingStates: Adding state machines to Java and the Swing toolkit. *Software: Practice and Experience*, 38(11):1149 – 1182, 2008.
3. C. Appert and S. Zhai. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proc. CHI '09*, 2289–2298. ACM.
4. O. Bau and W. E. Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proc. UIST '08*, 37–46. ACM.
5. J. Hong and J. Landay. Satin: a toolkit for informal ink-based applications. In *Proc. UIST '00*, 63–72.

BIOGRAPHIE

Caroline Appert a reçu le prix Gilles Kahn 2007 pour sa thèse "Modélisation, Évaluation et Génération de Techniques d'Interaction" soutenue au Laboratoire de Recherche en Informatique (Orsay) sous la direction de Michel Beaudouin-Lafon. Ses travaux visent à proposer des méthodes d'évaluation et de programmation pour faciliter l'introduction de techniques d'interaction innovantes dans les applications graphiques. Après un post-doctorat à IBM Almaden (San Jose, CA, USA), elle est actuellement chargée de recherche CNRS au LRI dans l'équipe InSitu.