Une architecture multi-dispositifs

Jean-Daniel Fekete, Pierre Dragicevic

UNE ARCHITECTURE MULTI-DISPOSITIFS

Jean-Daniel Fekete

Pierre Dragicevic

Ecole des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44307 Nantes Cedex 3
{fekete,dragice}@emn.fr

Résumé *— Les boîtes à outils graphiques ne gèrent qu'un clavier, une souris et un nombre limité de techniques d'interaction. La prolifération de nouveaux dispositifs, en particulier pour les jeux, la CAO, le dessin et l'accès aux handicapés, n'est donc pas prise en compte. Cet article décrit une nouvelle architecture permettant la gestion de multiples dispositifs d'entrée et leurs connexions avec des composants actifs. Notre architecture est loin d'être achevée et pose plus de questions qu'elle n'en résout, mais c'est aussi une bonne propriété pour un travail de recherche.*

.

1. Introduction

Les boîtes à outils graphiques supposent que l'application qu'elles servent à construire n'utilisera jamais qu'une souris, un clavier et un ensemble limité de techniques d'interaction. Cependant, plusieurs raisons nous poussent à diversifier les dispositifs d'entrée et les styles d'interaction (que nous appellerons ensemble méthodes d'interaction). Une première raison est la prolifération de nouveaux dispositifs d'interaction, due au développement d'applications très avides de ces dispositifs tels les jeux, la CAO, la modélisation 3D et l'illustration graphique [3]. Une seconde raison est que les interfaces graphiques ne sont plus limitées aux machines de bureau ou portables ; les palmtop et les ordinateurs vestimentaires utilisent aussi des interfaces graphiques, bien qu'avec des dispositifs autres qu'une souris et un clavier. La portabilité d'applications entre les machines de bureau et les machines nomades simplifierait considérablement la tâche des développeurs et des utilisateurs. Une troisième raison est l'utilisabilité par les handicapés, qui peuvent avoir besoin de méthodes d'interaction adaptées à leurs déficiences. Enfin, les applications fonctionnant sur des machines de bureau pourraient aussi bénéficier des nouvelles méthodes d'interaction comme la reconnaissance de geste, de parole ou de l'écriture manuscrite, pour ne citer qu'elles.

Cet article décrit une architecture de gestion des entrées pour de boîte à outils graphique interactive. Cette architecture est en cours d'implémentation mais certains de ses concepts ont été validés sur des prototypes.

Cette architecture put être qualifiée de multimodale dans un sens strict mais elle ne s'intéresse ni à la fusion de modalités [7], ni à l'interprétation sémantique de modalités comme la parole. Nous considérons plutôt cette architecture comme une infrastructure à la multimodalité avec fusion et fission.

2. Principes de l'architecture

Notre architecture repose sur le concept de dispositifs et de connexions. Un dispositif, au sens large, possède des canaux de sortie typés, ainsi que des canaux d'entrée. De plus, il peut effectuer des traitements internes. Une configuration d'entrée comprend un ensemble de dispositifs connectés. Par exemple, en suivant la taxonomie de dispositifs de la X Input Extension [6], on a les types suivants :

```
struct Button {
        boolean state ;
} ;
struct Locator {
        int x, y ;
} ;
device Mouse {
        output {
                Button button[5] ;
                Locator loc ;
        }
} ;
```

Les connexions permettent de construire une cascade de traitements. Par exemple, un dispositif clavier est une cascade partant d'une boîte à boutons (un bouton pour chaque touche), transformé en générateur de symboles, transformé en générateur de chaînes de caractères :

```
device RawKeyboard {
        output { boolean key[] ; }
} ;
```

Un clavier primitif est une boîte à boutons. Dans un deuxième temps, les événements touche sont sérialisés et un symbole est associé à chaque touche :

```
device SymKeyboard {
        input {
                boolean key[] ;
                Symbol mapping[] ;
        }
        output { Symbol sym ; }
} ;
```

```
device Keyboard {
        input {
                Symbol sym ;
                InputMethod method ;
        }
        output { String str ; }
} ;
```

Il faut connecter les canaux compatibles des RawKeyboard, SymKeyboard et Keyboard pour obtenir la fonctionnalité du clavier habituel, capable de générer des informations au niveau des touches concrètes, de leur valeur symbolique (flèche vers le haut, touche control ou touche A) ou de leur interprétation en caractères. La connexion entre les dispositifs se fait par des liens décrits explicitement.

Il s'agit donc d'un système à flot de données ou les dispositifs sont des unités de traitement, d'émission ou de réception de signaux typés.

3. Événements multi-dispositifs

Que produit une telle cascade de dispositifs ? Des événements ou des actions directes. Si la cascade doit s'arrêter et passer la main à une architecture de boîte à outils standard, alors un événement doit être produit. Dans notre architecture, un événement est un objet extrêmement simple qui ne contient que cinq champs : un temps, un identificateur unique, le pointeur vers le dispositif qui l'a émis, un chemin vers le canal de sortie qui a été modifié et qui a justifié la génération de cet événement, et enfin une liste d'événements de plus bas niveau qui sont à l'origine du déclenchement de celui-ci. Exprimé différemment, un événement indique un changement d'état dans les canaux de sortie d'un dispositif. Le champ temps indique le moment où ce changement a été constaté et le champ détails indique éventuellement le champ ou le groupe de champs concernés.

```
class Event {
        Time time :
        ID id ;
        Device device ;
        ChannelPath path ;
        Event dep[] ;
} ;
```

Le chemin permet de gérer différents niveaux de précision dans la notification. Par exemple, le clavier indique chacune de ses modifications touche par touche tandis que la souris indique que sa position a changé mais ne précise pas si c'est le canal x ou y qu a changé. Dans le premier cas, le chemin indiquera le canal précis du dispositif tandis que dans le second cas, il spécifiera le champ « loc » de la souris, sans plus de précision. Cette architecture impose un mécanisme d'introspection au niveau des dispositifs afin d'autoriser une application à analyser les capacités du dispositif. En cela, elle diffère de celle de Chatty [4], qui place l'introspection au niveau des événements.

4. Du dispositif au composant

Cette architecture permet de décrire les connexions des entrées jusqu'aux composants. Notre article [5] décrit une architecture MVC modifiée qui permet de gérer des dispositifs multiples dans une boîte à outils à composants. Nous avons défini des dispositifs « aiguillage » qui dirigent les valeurs et le contrôle en fonction de paramètres géométriques. Ces dispositifs prennent en entrée une table de régions et une position et en sortie une table de booléens indiquant, pour chaque région, si la position en entrée est dans la région. Ce dispositif permet donc d'inhiber ou d'autoriser le traitement des valeurs en fonction d'une position. Couplés à des dispositifs positionnels, ils permettent de construire un contrôle sur des composants. Ainsi, le composant « range slider » utilisé pour spécifier un intervalle de valeurs, en particulier pour les « dynamic queries and starfields display, est décrit en figure 1.

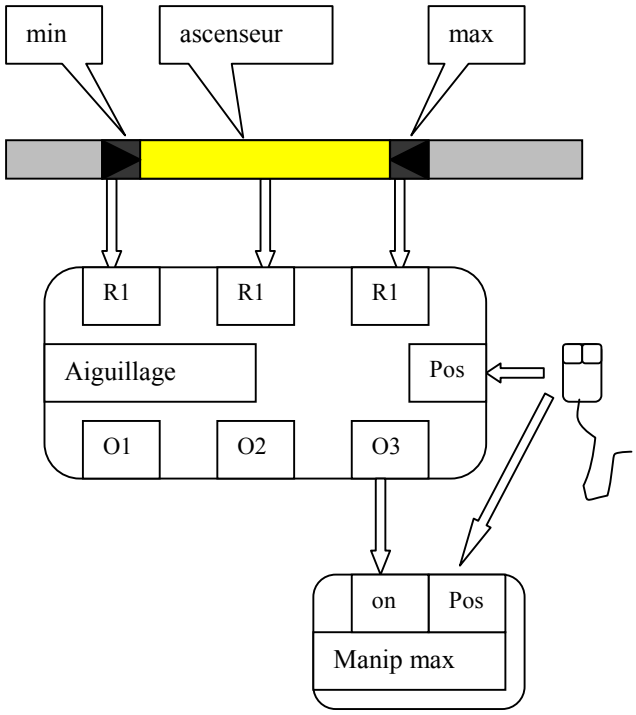

***Figure 1** — gestion d'un composant « range slider ». La manipulation est déclenchée par une position sur une région.*

Dans ce schéma, le composant RangeSlider est vu comme le dispositif suivant :

```
device RangeSlider {
        input {
                Manip min, max, thumb ;
        }
        output {
                Region min, max, thumb ;
        }
} ;
```

Un manipulateur est un objet, décrit dans [5] et capable de gérer une interaction continue en la décomposant en trois phases : début, manipulation et fin [2].

Avec notre modèle architectural, nous pouvons décrire par une construction à flot de données une configuration de connexions entre entrées et composants.

Nous réalisons actuellement un éditeur graphique de configuration pour faciliter la spécification d'un réseau de connexions.

5. Au-delà de la reconstruction

Nous avons jusqu'à présent décrit une configuration relativement standard, qui correspond à l'expression dans notre architecture de mécanismes implicites dans les boîtes à outils standard. Bien entendu, l'objectif de notre recherche est surtout d'exprimer des configurations non standards. Nous avons expérimenté les configurations suivantes :

- Utilisation de plusieurs dispositifs de bas niveau pour gérer trois pointeurs simultanés : la souris, une souris virtuelle pilotée à l'aide des flèches du clavier et souris virtuelle pilotée avec un système de reconnaissance vocale.
- Utilisation de reconnaissance de gestes.

Pour manipuler les gestes, nous définissons la structure de « trace » (stroke) qui est une liste de positions, éventuellement avec une référence temporelle :

```
struct Stroke {
        Position stroke[] ;
        Time time[] ;
} ;
```

Une trace peut être extraite de n'importe quel dispositif produisant des positions et peut être passé à des dispositifs faisant de l'affichage – pour l'écho lexical – ou du traitement, tel la reconnaissance de geste ou la reconnaissance de l'écriture manuscrite en ligne. La reconnaissance de geste [8] est un dispositif qui produit des symboles (un répertoire de mots) tandis que la reconnaissance d'écriture manuscrite en ligne produit des chaînes de caractères, compatible avec un clavier.

Notons qu'à partir d'une configuration existante, il est facile d'ajouter des dispositifs virtuels, par exemple pour stabiliser une souris dans le cadre de l'aide à certains handicaps telle la maladie de Parkinson.

6. Travaux futurs

Nous travaillons actuellement à la spécification formelle d'une sémantique pour ces réseaux d'interconnexion de dispositifs. Nous avons envisagé deux formalismes : les systèmes synchrones pour le contrôle et l'évaluation symbolique pour les valeurs.

En donnant une sémantique précise à ces réseaux, nous espérons pouvoir vérifier des propriétés élémentaires de comptabilité lors des connexions. Les types permettent déjà certaines vérifications, mais nous aurions besoin d'aller plus loin. Par exemple, nous aimerions pouvoir vérifier que tous les points de l'écran peuvent être atteints à partir d'un dispositif de pointage. Un joystick standard a deux axes dont les valeurs varient entre -128 et +127. Si on le connecte directement comme pointeur, on ne peut pas atteindre tous les points de l'écran (ou l'écran est très petit). Si on multiplie les coordonnées par un facteur constant, on ne peut pas non plus atteindre tous les points de l'écran. Si on utilise le joystick comme dispositif relatif, on peut atteindre tous les points de l'écran et, plus généralement, une infinité de valeur en un temps infini par intégration temporelle. C'est le type de propriétés que nous aimerions pouvoir exploiter.

En utilisant une sémantique proche du langage Esterel, nous pensons pouvoir, à terme, utiliser tous les outils de vérification proposés par Esterel [1] et toutes les recherches en cours autour du langage.

Bibliographie

[1] G. Berry. The Foundations of Esterel. In Proof, Language and Interaction: Essays in Honour of Robin Milner, G. Plotkin, C. Stirling and M. Tofte, editors, MIT Press, 1998.

[2] B. Buxton. Chunking and Phrasing and the Design of Human-Computer Dialogues, In Information Processing, Proceedings of the IFIP 10th World Computer Congress, pages 475-480. 1986.

[3] B. Buxton (Ed.) Interaction in 3D Graphics. Computer Graphics, vol. 32 N° 4. ACM.

[4] S. Chatty. Extending a Graphical Toolkit for two-handed Interaction. In Proceedings of the ACM CHI, page 195-203. Addison-Wesley. 1994.

[5] P. Dragicevic and J.-D. Fekete. Étude d'une Boîte à Outils Multi-Dispositifs. Actes des 11ième journées francophones sur l'interaction Homme-Machine IHM 99, pages 55-62, Cépaduès 1999.

[6] P. Ferguson. The X11 Input Extension: A tutorial. The X Resource, vo. 4 n° 1 pp. 171-194.

[7] L. Nigay and J. Coutaz. A design space for multimodal systems: concurrent processing and data fusion. In Proceedings of the ACM CHI, pages 172-178. Addison-Wesley. 1993.

[8] D. Rubine, Specifying Gestures by Example. In Proceedings of SIGGRAPH . , pages 329-337. 1991.