

Flexibilité et modularité pour la conception d'interactions: Le modèle d'architecture logicielle des Graphes Combinés

Stéphane Huot, Pierre Dragicevic, Cédric Dumas

▶ **To cite this version:**

Stéphane Huot, Pierre Dragicevic, Cédric Dumas. Flexibilité et modularité pour la conception d'interactions: Le modèle d'architecture logicielle des Graphes Combinés. 18ème conférence francophone sur l'Interaction Homme-Machine, IHM'06, Association Francophone d'Interaction Homme-Machine, 2006, Montréal, Canada. pp.43-50, 10.1145/1132736.1132742 . inria-00550596

HAL Id: inria-00550596
https://inria.hal.science/inria-00550596

Submitted on 29 Dec 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Flexibilité et modularité pour la conception d'interactions : Le modèle d'architecture logicielle des Graphes Combinés

Stéphane Huot
Ecole Nationale Supérieure des Télécommunications
46, rue Barrault
75634 Paris Cedex 13, France
Stephane.Huot@enst.fr

Pierre Dragicevic
Intuilab
Prologue 1
La Pyrénéenne
31672 Labège, France
dragice@intuilab.com

Cédric Dumas
Ecole des Mines de Nantes
La Chantrerie
4, rue Alfred Kastler
44307 Nantes, France
Cedric.Dumas@emn.fr

RESUME
Cet article présente le modèle d'architecture logicielle des *Graphes Combinés* qui a servi de base à l'implémentation de la boîte à outils post-WIMP MAGGLITE. Ce modèle étend et affine l'architecture graphique des *graphes de scène* en décrivant l'interaction séparément avec des *graphes d'interaction*. Les graphes sont combinés dynamiquement à l'exécution de l'application, grâce à des modules de communication spécifiques : les *Points d'Accès aux Interactions*. Il s'en suit une description flexible et modulaire des graphismes et des interactions, avantageuse du point de vue du prototypage, de l'implémentation et de l'utilisation de périphériques d'entrée et de techniques d'interaction non standard.

MOTS CLES : Architecture logicielle, Boîtes à outils d'IHM, ICON, MAGGLITE, techniques d'interaction.

ABSTRACT
This paper presents the *Mixed Graphs* software architecture of the MAGGLITE post-WIMP toolkit. This model extends and refines the *scene graph* architecture by describing interactions with another structure: *interaction graphs*. Graphs are dynamically combined at runtime, thanks to specific communication components named *Interaction Access Points*. This adaptable and modular approach for describing interactive graphics is gainful for prototyping, implementing and using alternative input devices and/or interaction techniques.

CATEGORIES AND SUBJECT DESCRIPTORS: H5.2 [Information Interfaces]: User Interfaces; D2.11 [Software Engineering]: Software Architectures.

GENERAL TERMS: Design, Human Factors.

KEYWORDS: GUI architectures, GUI toolkits, ICON, MAGGLITE, interaction techniques.

INTRODUCTION
La plupart des outils logiciels de conception d'interfaces reposent sur des composants monolithiques qui encapsulent présentation et interactions. Avec de tels modèles, prototyper des interfaces et des interactions innovantes est un processus long et coûteux, plus à la portée des spécialistes de la programmation que des concepteurs d'IHMs [9]. Bien que de nombreux travaux aient déjà permis des avancées importantes, il est encore nécessaire d'améliorer les outils de prototypage et de conception d'interactions non-standard. Nous pensons que cette amélioration passe en partie par des modèles d'architecture logicielle offrant toujours plus de flexibilité et de modularité, afin de permettre la réalisation d'outils de prototypage et de conception plus performants et surtout innovants.

Une avancée majeure dans les boîtes à outils pour l'interaction non-standard a été d'utiliser la structure de graphe de scène pour décomposer les widgets monolithiques en objets graphiques de granularité plus fine (Jazz/Piccolo [6], CPN Tools [2], UBIT [18], DoPIdom [4], INDIGO [7]). Ces approches ont permis d'optimiser ou de distribuer le rendu graphique, et d'améliorer dans une certaine mesure l'extensibilité des bibliothèques de widgets. En séparant et décomposant également les comportements, les approches comme DoPIdom et INDIGO ont permis d'obtenir encore plus de flexibilité et d'évolutivité.

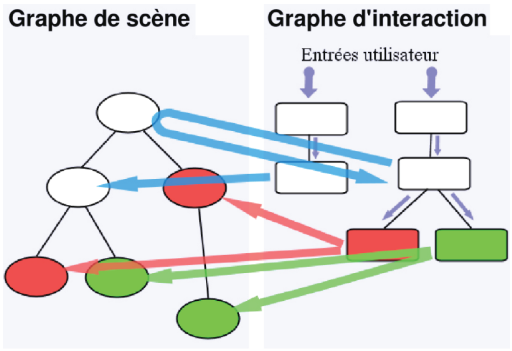


Figure 1: Un graphe de scène et un graphe d'interaction sont combinés à l'exécution. Certains nœuds du graphe d'interaction communiquent avec des nœuds compatibles du graphe de scène.

Le modèle des *graphes combinés* que nous décrivons ici pousse cette philosophie plus loin en décomposant encore plus finement la gestion des sorties (graphe de scène) et celle des entrées (interactions et comportements), et en

décrivant leurs relations. Cette séparation passe par l'utilisation d'un modèle plus flexible et modulaire que le modèle traditionnel à événements : les *graphes d'interaction*, basés sur l'assemblage de modules de flot de données. Ces derniers manipulent le graphe de scène et ses éléments de manière dynamique (voir *Figure 1*), en réponse à des actions de l'utilisateur ou des événements système. Et comme les graphes de scènes séparent la partie graphique en composants primitifs, les graphes d'interaction décomposent la structure souvent complexe et monolithique des architectures à événements.

Cette architecture a été utilisée pour concevoir MAGGLITE [14], un environnement de développement adapté au prototypage d'applications post-WIMP ou d'applications standard proposant des interactions avancées. Les graphes d'interaction peuvent y être réassemblés graphiquement de la même manière que les graphes de scène peuvent être modifiés avec des outils auteurs. Il est ainsi possible de recomposer les associations vue/contrôleur pour adapter l'interface ou tester des techniques d'interaction alternatives sans avoir à programmer, ni relancer l'application ; une telle approche dynamique autorise des itérations rapides encourageant fortement l'exploration. Dans cet environnement, il est aussi simple pour le développeur d'implémenter de nouvelles interactions, que pour le concepteur d'interfaces de les assembler. L'utilisateur final peut aussi, dans une certaine mesure, adapter l'interface et les interactions à ses besoins ou possibilités.

Après une brève description de l'architecture des graphes de scène, nous introduirons les *graphes d'interaction* en décrivant plus particulièrement les « charnières » entre les entrée et les sorties : les *Points d'Accès aux Interactions*, nœuds chargés de la liaison dynamique avec le graphe de scène. Nous y ferons aussi apparaître les principes et les mécanismes de base de la boîte à outils MAGGLITE, afin d'en cerner la philosophie et les apports.

GRAPHES DE SCENE

Les graphes de scène, introduits dans les boîtes à outils 3D du milieu des années 1990 [20], reposent sur une structure de graphe acyclique orienté pour organiser hiérarchiquement une scène graphique. Chacun des nœuds de cette arborescence est un élément de la scène à représenter, pouvant lui-même être un graphe de scène.

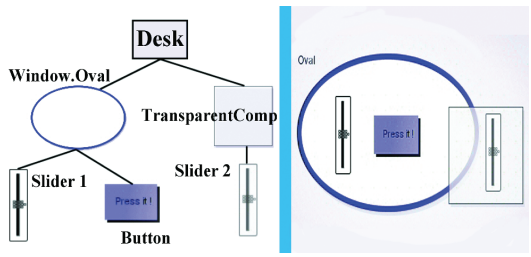


Figure 2 : Un graphe de scène et l'interface correspondante.

Cette représentation a fait des graphes de scène un standard d'implémentation des boîtes à outils ou logiciels pour la 3D. Elle est maintenant aussi adoptée dans les boîtes à outils 2D (voir *Figure 2*), spécialement dans celles dédiées à l'interaction avancée. Si nous avons choisi cette structure, c'est avant tout pour sa modularité, propice à la même décomposition de la description de l'interaction et surtout à son externalisation.

L'interaction dans les graphes de scène

Dans les approches standard, l'interaction est décrite par des nœuds spécifiques du graphe de scène, quand elle n'est pas tout simplement encapsulée dans les nœuds graphiques. Par exemple, les boîtes à outils Jazz et son évolution Piccolo sont basées sur une structure très complète et avancée de graphes de scène [6]. Par contre, les interactions sont décrites au niveau des nœuds (objets graphiques) en s'appuyant sur l'architecture à évènements peu évolutive de la librairie Swing. De fait, même si cette boîte à outil offre une grande latitude pour la création de nouveaux composants graphiques, elle reste relativement fermée pour ce qui est du prototypage d'interactions avancées. Outre l'aspect statique d'une telle approche, qui impose de « brancher », et donc figer, les interactions à la création du programme, se pose aussi le problème de l'extensibilité : l'introduction d'interactions avancées, de périphériques nouveaux et leur réutilisation dans d'autres contextes reste laborieuse.

Plusieurs approches ont déjà permis d'atteindre un plus haut niveau de flexibilité en externalisant l'interaction : INDIGO [7], qui utilise des machines à états hiérarchiques pour décrire l'interaction séparément du graphe de scène et DoPIdom [4] qui associe les nœuds graphiques à des composants interactifs liés par un schéma producteur/consommateur. Notre modèle de graphes combinés s'appuie sur un autre formalisme pour décrire les interactions : les graphes d'interaction.

GRAPHES D'INTERACTION

Les graphes d'interaction reposent sur le modèle de flot de données ICOM (Input Configuration Model) de la boîte à outils ICON [12]. Ses *configurations d'entrée* sont composées de modules de traitement (*dispositifs*) qui permettent, en les connectant entre eux par des canaux typés (*slots*), de transformer et d'utiliser dans des applications les données reçues par des périphériques d'entrée (voir *Figure 6*). La boîte à outil ICON gère un grand nombre de périphériques standard ou avancés, fournit des dispositifs de traitement des données ainsi qu'un configurateur graphique interactif (Input Configurator) pour construire des configurations d'entrée de manière dynamique, sans avoir à relancer l'application.

Dans notre architecture, un graphe d'interaction est donc une configuration d'entrée d'ICON qui décrit les comportements du graphe de scène et de ses objets, ainsi que les manipulations possibles par un utilisateur. Les figures *3*, *4* et *5* présentent la construction d'un graphe d'interaction qui décrit la manipulation à la souris des objets présents dans un graphe de scène :

1. *Figure 3* - Le dispositif *souris* est connecté à un dispositif *curseur* par l'intermédiaire d'adaptateurs (valeurs relatives à valeurs absolues). Chaque déplacement de la souris déplace un pointeur à l'écran.

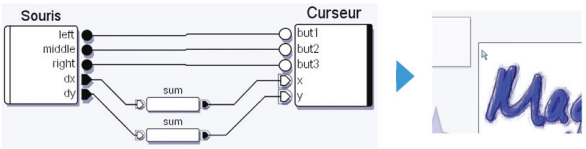


Figure 3 : Connexion de la souris à un curseur.

2. *Figure 4* - Le dispositif *curseur* est connecté ensuite à un dispositif *pick* qui permet de sélectionner les objets graphiques (nœuds du graphe de scène) qui se trouvent sous le curseur.

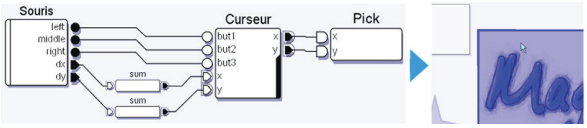


Figure 4 : Connexion d'un dispositif de sélection (*pick*).

3. *Figure 5* - Enfin, le dispositif *pick* et un bouton de la souris sont connectés à un dispositif *drag* afin de déplacer les objets sélectionnés qui le permettent.

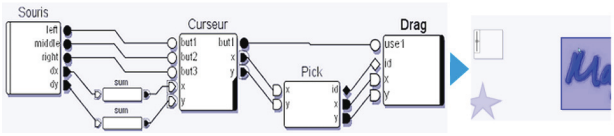


Figure 5 : Connexion d'un dispositif d'interaction (drag).

Des graphes plus complexes peuvent bien évidemment être décrits. Le graphe d'interaction de la *Figure 6* a été construit pour manipuler le graphe de scène de la *Figure 2*.

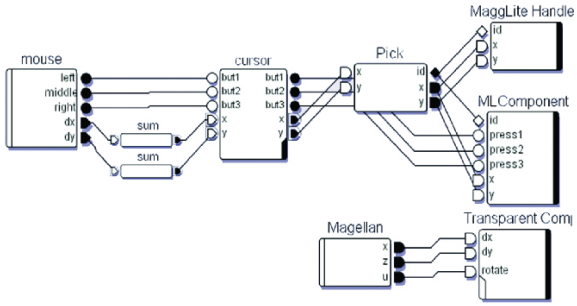


Figure 6 : Un graphe d'interaction possible pour le graphe de scène de la *Figure 2*.

La souris contrôle un curseur et un dispositif de sélection qui transmet les objets sélectionnés à deux dispositifs représentant des techniques d'interactions. Un périphérique d'entrée est connecté à un dispositif représentant un objet graphique du graphe de scène (*TransparentComp*).

Ces exemples illustrent l'utilisation des dispositifs définis dans le modèle ICOM: les *dispositifs système* (souris, Magellan) et les *dispositifs de la bibliothèque* (curseur, adaptateurs). Pour les dispositifs qui permettent de contrôler l'application (sélection, glisser-déposer, etc.), le modèle ICOM définit la classe des *dispositifs d'application*. Ce sont les points d'entrée de l'application, lien entre ses outils et la configuration d'entrée.
Pour associer configurations d'entrée et graphes de scène, nous avons généralisé cette notion de dispositifs d'application en les décrivant au niveau des graphes combinés comme le lien entre les graphes de scène et d'interaction : ce sont les *Points d'accès aux interactions*.

POINTS D'ACCES AUX INTERACTIONS (IAPS)

Les Points d'Accès aux Interactions (IAPs pour « Interaction Access Points ») sont les nœuds spécifiques du graphe d'interaction qui établissent des points de connexion dynamiques entre le graphe d'interaction et le graphe de scène (dans la *Figure 1*, ce sont les nœuds du graphe d'interaction d'où partent les flèches). Les IAPs transmettent et reçoivent des données du graphe de scène, modifient sa structure en insérant ou supprimant des nœuds, changent les propriétés ou déclenchent des comportements spécifiques aux nœuds. Ils représentent différentes manières de lier des dispositifs d'entrée aux composants graphiques du graphe de scène, construisant les charnières entre les entrées et les sorties de l'application. Nous distinguons quatre classes de IAPs :

1. les *dispositifs d'interaction*, pour manipuler toute une classe d'objets graphiques ;
2. les *manipulateurs*, pour contrôler une instance, un nœud particulier du graphe de scène ;
3. les *dispositifs de comportements*, pour spécifier le comportement d'un ou plusieurs objets graphiques ;
4. les *outils internes*, pour réaliser une action dans l'espace d'un objet graphique.

Dispositifs d'interaction

Les dispositifs d'interaction définissent les fonctionnalités d'une technique d'interaction applicable à une classe de nœuds du graphe de scène, et donc à une classe particulière de composants de l'interface (voir *Figure 7*).

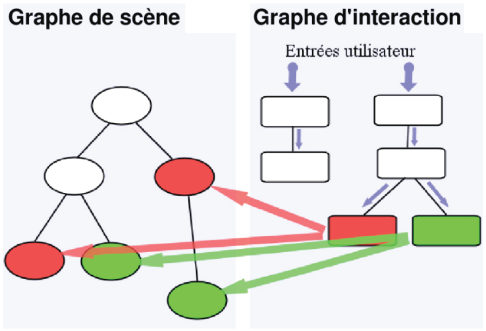


Figure 7 : Les dispositifs d'interaction communiquent avec une classe de nœuds du graphe de scène.

Conventionnellement, les techniques d'interactions sont décrites de façon abstraite et générique, sans les lier à un périphérique d'entrée et à une tâche (ou un objet) spécifiques (c'est le style d'interaction dans la *Figure 8*). Pourtant, leur implémentation concrète s'avère être fortement dépendante du couple {dispositif, tâche}, de par la rigidité des outils actuels. Il est alors souvent fastidieux d'adapter une technique d'interaction à une autre classe d'objets et à d'autres périphériques d'entrée que ceux pour lesquels elle a déjà été implémentée.

ICON a permis de séparer la gestion des entrées de la réalisation des interactions, introduisant la notion d'adaptabilité en entrée [12]. Notre modèle permet cette même adaptabilité et flexibilité au niveau de l'interaction et de son objet d'intérêt en externalisant le style d'interaction avec les dispositifs d'interaction (voir *Figure 8*).

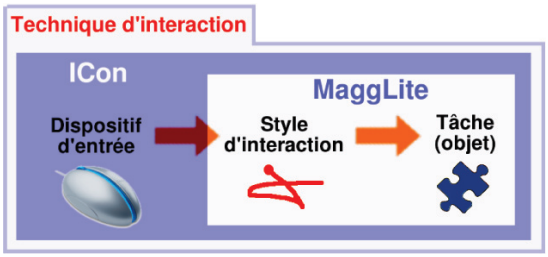


Figure 8 : Une technique d'interaction est définie par un dispositif d'entrée, un style d'interaction, et une tâche à réaliser.

Utilisation et exemple. Le champ d'application des dispositifs d'interaction est vaste. Ils permettent de factoriser et de réutiliser le noyau d'un style d'interaction. Par exemple, le graphe d'interaction de la *Figure 5* décrit la technique du Glisser-déposer à partir du dispositif *drag*, et permet de l'utiliser sur tous les objets compatibles. Dans la configuration proposée, la manipulation sera réalisée en se plaçant sur l'objet, en pressant le bouton gauche de la souris et en la déplaçant. Mais cette technique n'est pas figée et peut être initiée par d'autres dispositifs (clavier ou commande vocale pour le déclenchement, des dispositifs absolus, relatifs ou même une Webcam pour le déplacement) ou remplacée par une autre.

Fonctionnement. Les dispositifs d'interaction reçoivent les références des objets graphiques avec lesquels ils vont devoir coopérer par l'intermédiaire du flot de données (le slot d'entrée *Id* dans la *Figure 5*, généralement connecté au dispositif de sélection *pick*). Les classes de composants graphiques déclarent leurs capacités et les dispositifs d'interaction sont spécialisés dans la gestion d'une ou plusieurs capacités particulières. Par exemple, une classe d'objets graphiques pouvant être déplacés signale cette capacité (par l'interface *Moveable* dans notre implémentation Java) : toutes ses instances seront compatibles avec les dispositifs d'interaction permettant le déplacement d'objets. Ces dispositifs représentent des implémentations d'*interactions génériques*, n'étant pas liées aux objets eux-mêmes mais à leurs caractéristiques et possibilités.

Cette modularité rend les interactions utilisables dans toutes les applications et adaptables dynamiquement à tous types de périphériques pour composer de nouvelles techniques d'interaction. Elles sont aussi implicitement étendues à d'autres classes d'objets graphiques (donc de tâches) si celles-ci implémentent les capacités idoines. MAGGLITE fournit essentiellement des dispositifs de manipulation directe tels que le dispositif *drag* vu précédemment. Mais cette architecture permet aussi la réalisation de techniques de visualisation d'informations (Fisheye Lens), de méthodes de saisie de texte, etc.

Manipulateurs

Contrairement aux dispositifs d'interaction, qui peuvent potentiellement agir sur n'importe quel objet compatible du graphe de scène, les manipulateurs sont des dispositifs IAP liés à une instance, un nœud particulier du graphe de scène (voir *Figure 9*). Ils déclenchent des commandes que sait exécuter l'unique objet graphique qu'ils contrôlent : se déplacer, se redimensionner, etc.

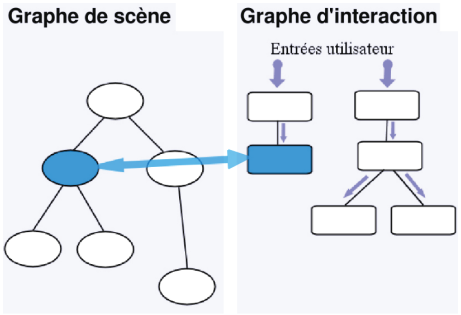


Figure 9 : Un manipulateur communique avec un nœud spécifique et unique du graphe de scène.

Exemple. La *Figure 10* présente la connexion d'un dispositif d'entrée à 6 degrés de liberté (Magellan) au manipulateur d'un objet graphique (un rectangle contenant des vues miniatures de calques de dessin). Les axes *x* et *z* du dispositif physique sont connectés aux slots de déplacement relatif de l'objet (*dx* et *dy*) et son axe de rotation *u* au slot de rotation de l'objet (*rotate*). Les actions sur ces axes permettent donc d'agir directement sur l'objet graphique (déplacer l'ensemble des vues miniatures). Les signaux de chaque canal transitent par les dispositifs de la configuration (le graphe d'interaction) et sont transmis à l'objet (le graphe de scène) par son manipulateur (voir *Figure 10*). Ce mécanisme est transparent du point de vue de l'utilisateur pour qui le périphérique physique est directement couplé à l'objet graphique. Ce principe d'association directe est une manière simple et efficace de faire de la manipulation directe et de prototyper par exemple des interactions bimanuelles, renforçant de plus la notion de contrôle dans l'approche en flot de données.

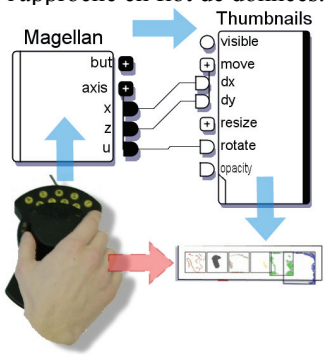


Figure 10 : 3 axes d'un dispositif Magellan sont connectés aux slots déplacement/rotation du manipulateur d'un objet graphique (la barre de vues miniatures).

Fonctionnement. Dès qu'un objet graphique est ajouté au graphe de scène, il apparaît sous la forme d'un dispositif dans la bibliothèque du configurateur graphique d'ICON : c'est un manipulateur, qui externalise les points d'entrée et de sortie de l'objet. Pour reprendre l'exemple d'un objet qui peut être déplacé, son manipulateur présentera des slots d'entrée *move*, prévus pour recevoir les valeurs de déplacement (absolues ou relatives). Les manipulateurs peuvent aussi présenter des slots de sortie, correspondants à des états que l'objet graphique peut transmettre (par exemple, tous les manipulateurs ont un slot de sortie « *Id* » qui transmet la référence de l'objet graphique qu'ils représentent). Enfin, ils sont générés dynamiquement (par

introspection) lors de l'instanciation des objets graphiques et ajoutés à la bibliothèque des dispositifs du configurateur graphique pour être utilisés dans la construction des graphes d'interaction.

Dispositifs de comportements

Les dispositifs de comportement décrivent des comportements ou des états génériques (pulsation, surlignage, etc.), applicables à n'importe quel nœud ou groupe de nœuds compatibles du graphe de scène. Ils se connectent par l'intermédiaire des manipulateurs (voir *Figure 11*). Ainsi, il est possible de spécifier lors de l'exécution de l'application quel va (vont) être le(s) comportement(s) graphique(s) d'un objet de l'interface.

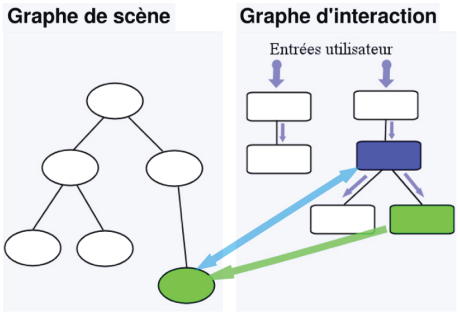


Figure 11 : Dispositifs de comportement. Un dispositif de comportement est connecté en aval d'un manipulateur afin de communiquer avec le nœud correspondant du graphe de scène.

Ces dispositifs externalisent les comportements des objets graphiques pour les réutiliser simplement et dynamiquement sur tout type de composants, et rendent leur déclenchement complètement configurable. Par rapport aux architectures logicielles standard, qui proposent différentes classes d'objets graphiques pour différents comportements ou à l'inverse une seule classe monolithique regroupant tous les comportements [6], notre proposition permet d'associer dynamiquement objet(s) et comportement(s) à un niveau de granularité plus fin.

Utilisation et exemples. Dans la *Figure 12*, un objet graphique « *Etoile* » est connecté, par l'intermédiaire du slot de sortie « *Id* » de son manipulateur, à un comportement de pulsation. L'activation de ce dernier est commandée par un slot de sortie du manipulateur, indiquant si l'objet est sélectionné ou non. Ainsi, lorsque l'objet est sélectionné (à l'aide du dispositif *pick*), le comportement est activé et l'objet émet une pulsation.

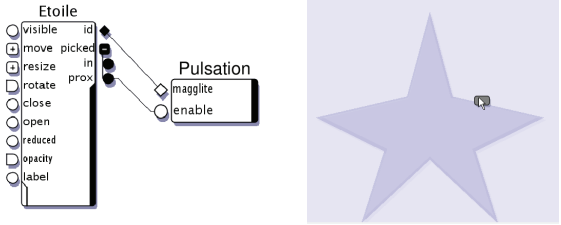


Figure 12 : L'objet « *Etoile* » est connecté au comportement « *pulsation* », déclenché lorsque l'objet est sélectionné.

La flexibilité des graphes d'interaction et de l'approche en flot de données permet bien sur de contrôler le déclenchement du comportement par un évènement non directement lié à l'objet associé (voir *Figure 13*) : un périphérique d'entrée, un manipulateur d'un autre objet graphique ou tout autre dispositif compatible.

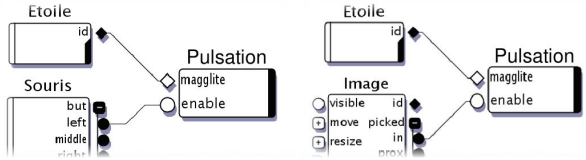


Figure 13 : Le comportement est déclenché par un périphérique ou par un état d'un autre objet graphique.

Enfin, un objet graphique peut être connecté à plusieurs comportements, plusieurs objets à différentes instances d'un comportement (même comportement, déclenchement différent) et une même instance d'un comportement peut être associée à plusieurs objets grâce à un dispositif *groupe* (même comportement et déclenchement).

Fonctionnement. Les dispositifs de comportement présentent deux entrées par défaut (voir *Figure 12*) : un slot *magglite* (pour connecter le(s) manipulateur(s) et obtenir par le flot de données la ou les références à l'objet ou groupe d'objets concerné) et un slot *enable* (active ou désactive le comportement). A la réception d'une valeur « vrai » par son slot d'activation, le dispositif va appliquer le comportement pour lequel il a été programmé à l'objet ou groupe d'objets graphique(s) connecté(s). Il est évident que d'autres slots peuvent être ajoutés dans le cas de comportements plus complexes que ceux proposés à titre d'exemple (des comportements liés à des entrées ou des actions décrites dans le graphe d'interaction).

Outils Internes

Les outils internes permettent d'accomplir une tâche dans un composant graphique spécifique. Leur action sur le graphe de scène est donc similaire à celle des manipulateurs, n'envoyant des messages qu'à un seul nœud du graphe de scène. Toutefois, alors qu'un manipulateur est la représentation d'un objet et des actions qu'il sait réaliser (déplacement, rotation, activation d'états, etc.), les outils internes sont des actions applicables uniquement dans l'objet graphique, conformément aux tâches qu'il permet d'accomplir (le dessin, par exemple).

Utilisation et exemples. Les outils internes permettent de remplir des tâches positionnelles et doivent donc recevoir en entrée des valeurs de position (en coordonnées écran). Deux méthodes de traitements sont alors possibles, caractérisant deux modes: le *mode local* et le *mode global*. Nous prendrons pour exemple de description de ces modes un outil interne de dessin, permettant de tracer des traits sur le composant auquel il est ajouté.

Mode local. Les valeurs reçues par les slots d'entrées de coordonnées sont converties dans le système de coordonnées local de l'objet graphique auquel est attaché l'outil. Ce mode permet une projection complète de la portée du périphérique d'entrée connecté sur la surface visible du composant. Dans la *Figure 14*, l'outil interne de dessin est connecté à une tablette graphique en mode local, approprié aux périphériques d'entrée absolus.

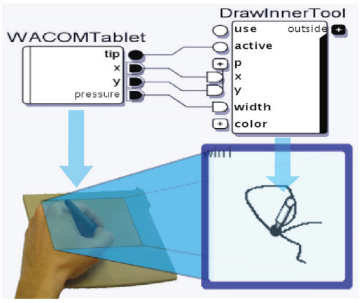


Figure 14 : Outil Interne en mode local.

Mode global. Dans ce mode, l'outil n'opère aucune conversion sur les coordonnées écran. Ainsi, la portée du dispositif d'entrée est projetée sur l'écran. L'outil n'est actif que si les coordonnées reçues sont dans les limites de l'objet graphique auquel il est attaché. Dans le cas contraire, l'outil est inactif et les coordonnées sont transmises par ses slots de sortie. Dans la *Figure 15*, deux outils internes sont connectés en mode global. Le premier dans la chaîne reçoit ses valeurs d'entrée d'un dispositif tablette graphique. Il est attaché au composant à bord bleu et son action est de reconnaître et interpréter des gestes. Le second outil est un outil interne de dessin, attaché au bureau de l'application. Ses slots d'entrée sont connectés aux slots de relais de l'outil de geste.

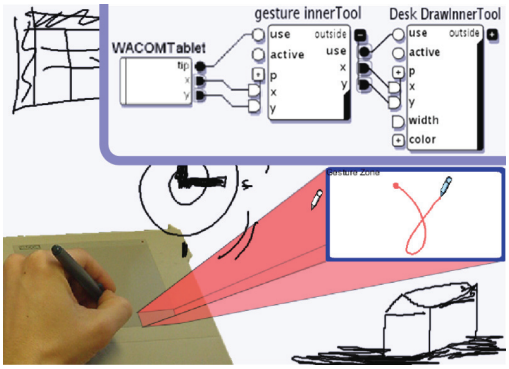


Figure 15 : Outil Interne chaînés en mode global.

Lorsque les coordonnées reçues sont dans les frontières du composant de reconnaissance de geste, l'outil de geste est activé et ne transmet aucune valeur à l'outil de dessin du bureau. Si les coordonnées sont en dehors des frontières de la zone de gestes, l'outil de gestes est désactivé et il transmet les coordonnées. L'utilisateur peut donc dessiner n'importe où sur le bureau, excepté dans la zone de gestes où ses traits sont interprétés. Cette projection de zones de l'écran sur les dispositifs d'entrée est une manière efficace de construire des interactions post-WIMP, tout spécialement avec des dispositifs tels que des tablettes, tablettes écran ou écran tactiles.

Fonctionnement. Les outils internes doivent être assignés (par programmation) à un nœud compatible du graphe de scène et n'agissent alors qu'à l'intérieur du contour graphique de celui-ci. Ils procurent un retour visuel de type curseur qu'ils se chargent d'installer dans le graphe de scène, en amont du composant graphique. Proches des *« Local Tools »* [5], alternative aux palettes d'outils, les outils internes offrent en plus la possibilité d'être associés librement à de nombreux périphériques d'entrée et assignés à l'espace d'un composant graphique.

DISCUSSION

Cette dernière partie met en relation notre modèle concret avec le modèle d'interaction de l'Interaction Instrumentale et le modèle MVC. Enfin, nous positionnons notre approche par rapport à des travaux similaires.

IAPs et l'interaction instrumentale

Considéré comme une avancée majeure dans la modélisation des interfaces post-WIMP, le modèle de l'interaction instrumentale [3] étend et généralise le modèle de la manipulation directe en introduisant les « instruments d'interaction » comme moyen d'interagir avec les objets du domaine. L'architecture de MaggLite s'applique bien à ce modèle d'interaction. Les IAPs s'intègrent dans la description d'instruments d'interaction par leur association avec des périphériques d'entrée. Ils définissent, avec certaines parties du graphe de scène (retour graphique ou vue de l'instrument), la composante logique des instruments (voir Figure 16).

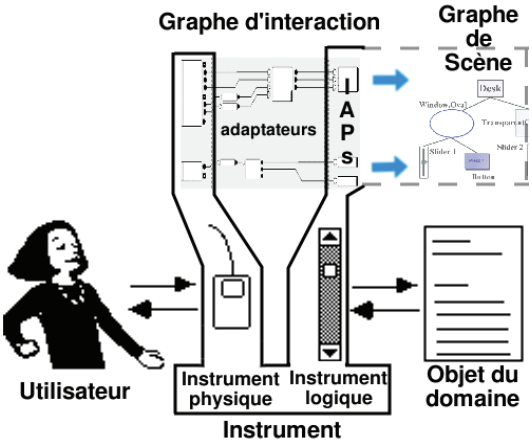


Figure 16 : IAPs et Interaction Instrumentale.

De plus, les IAPs observent le principe de réification : ils réifient les commandes qui contrôlent soit l'objet qu'ils représentent (manipulateurs), soit la classe d'objets avec lesquels ils sont compatibles (dispositifs d'interaction et outils internes). Par exemple, le dispositif *drag* connecté à un dispositif d'entrée et un curseur compose un instrument qui réifie le concept de déplacement des objets. Il est aussi polymorphe [3], car il peut être utilisé avec tous les objets pour lesquels il a du sens, mais aussi parce qu'il peut être adapté à d'autres contextes en utilisant diverses méthodes d'entrée (périphériques, reconnaissance vocale, etc.). Enfin, les manipulateurs permettent la manipulation directe des objets avec un instrument physique (périphériques d'entrée) : l'interaction est directe, physique, et le contrôle totalement explicite.

Notre modèle d'implémentation sépare donc explicitement les instruments physiques (périphériques d'entrée) des instruments logiques (dispositifs d'interaction, outils internes et manipulateurs) et permet leur association dynamique pour créer des instruments. Cette granularité très fine est un atout pour respecter les principes de

conception qui découlent de l'interaction instrumentale (réification, polymorphisme et réutilisabilité), facilitant ainsi le prototypage d'instruments basés sur un grand nombre de techniques d'interaction mais aussi leur intégration dans une même application.

MVC

MVC [17] est un modèle devenu référence dans l'implémentation des applications interactives. Ces dernières sont décomposées en agents MVC, chacun présentant un Modèle, une ou plusieurs Vues et un ou plusieurs Contrôleurs. Ce modèle promet une certaine indépendance entre l'aspect (sorties) et le comportement des objets graphiques (entrées). Toutefois, sa mise en œuvre se révèle complexe et une grande majorité des boîtes à outils fusionnent la vue et le contrôleur dans leurs implémentations [11]. Elles assurent la flexibilité au niveau modèle, mais gardent une approche monolithique du point de vue de l'interaction (en entrée et en sortie).

Les configurations d'entrée d'ICON représentent le Contrôleur. L'architecture des graphes combinés place la Vue au niveau des objets graphiques du graphe de scène, le Modèle étant décrit dans l'application. Pour les interactions, ce sont les Points d'Accès à l'Interaction qui vont établir la connexion entre les différents composants de MVC, de manière similaire au contrôleur de PAC [10]. Cette externalisation de ces derniers et leur connectivité dynamique est un apport important de notre architecture logicielle. Elle est plus flexible que beaucoup d'autres approches basées sur MVC, que ce soit pour le développement d'interfaces et la description d'interactions, mais aussi pour l'extensibilité de la boîte à outils.

Boîtes à outils avancées

Dans la même optique de séparation entre graphismes et interactions, les boîtes à outils Intuikit [9] et INDIGO [7] permettent de décrire une partie des interactions par des machines à états, indépendamment de la structure de graphe de scène. Si ces approches sont bien adaptées à la description des aspects discrets de l'interaction (par exemple, changements de mode sur des événements de type « clic »), elles sont moins explicites sur ses aspects continus (par exemple, reconnaissance de gestes sur des événements positionnels), pourtant prédominants dans les interfaces post-WIMP [16].

DoPIdom [4] propose un modèle centré sur les documents où les objets graphiques sont explicitement liés à des composants interactifs qui interagissent avec des instruments d'interaction selon un schéma producteur/consommateur. Les données du domaine sont nettement séparées de leur présentation, mais les comportements sont définis par des composants ad hoc de granularité moins fine que ceux des graphes d'interaction.

Les IAPs peuvent aussi être rapprochés des « interacteurs » de GARNET [19], bien que ces derniers soient écrits pour une souris et un clavier, et que la nature de leur association avec les objets graphiques soit figée.

Enfin, ces environnements ne proposent actuellement pas d'outils de prototypage interactifs et de programmation visuelle, paradigme auquel notre approche est particulièrement adaptée car proposant un seul et même modèle de la gestion des dispositifs d'entrée jusqu'aux objets du graphe de scène (le flot de données permet de visualiser « ce qui se passe » : la transmission et le traitement des données lors de l'interaction).

Editeurs visuels d'interaction

Depuis Thinglab [8], éditeur basé sur la résolution de contraintes et qui a permis de spécifier certains comportements interactifs, des outils visuels comme PetShop [1] (éditeur de réseau de Pétri) décrivent les parties orientées contrôle et hautement modales des applications interactives. Quand aux éditeurs de flot de données actuels, ils s'adressent surtout à des tâches spécifiques telles que le contrôle d'instruments de mesure ou la création musicale. Même si de telles approches ont été proposées pour contrôler l'interaction (notamment en 3D) ou décrire des animations [13], elles restent très spécialisées et n'offrent pas autant de possibilités que ICON et MAGGLITE (dispositifs et pointeurs multiples, interactions avancées, etc.).

Toutefois, l'approche à flot de données possède aussi ses limites, en particulier pour décrire des comportements faisant un usage important de multiplexage temporel et des modes d'interaction. Bien que cela ne pose pas de problème au niveau conceptuel, il s'avère que les graphes d'interaction deviennent complexes et leur programmation visuelle peu intuitive. Cet écueil peut être contourné par une approche totalement instrumentée (un périphérique par instrument). Mais une solution plus complète consisterait justement à intégrer une approche orientée contrôle (comme le formalisme ICO [1] utilisé dans PetShop, ou les machines à états hiérarchiques utilisées dans INDIGO) pour spécifier plus finement et interactivement la communication entre les IAPs et les objets graphiques du graphe de scène. Réunir ces deux approches au niveau des IAPs pourrait être une solution aux problèmes de l'unification des formalismes de contrôle et de flot de données dans un même langage visuel [16].

Les IAPs: le cœur des graphes combinés

Finalement, les graphes combinés reposent essentiellement sur la séparation des entrées et des sorties en deux graphes distincts, par l'intermédiaire des modules de combinaison dynamique : les *Points d'Accès à l'Interaction*. Les quatre classes de IAPs permettent de définir :

1. des actions génériques sur une classe de composants graphiques (les dispositifs d'interaction);
2. des actions particulières à un composant graphique (les manipulateurs);
3. des comportements graphiques (les dispositifs de comportement);
4. des outils propres à un composant graphique (les outils internes).

Ces abstractions établissent différents modes d'accès à un graphe de scène pour l'interaction (par classes ou instances de nœud) et permettent de concrétiser un grand nombre de techniques d'interaction. Mais il serait toute-

fois ambitieux de notre part d'affirmer que les IAPs couvrent tous les besoins en terme de description des interactions. Ils offrent avant tout un cadre extensible pour l'exploration, permettant la réunion de nombreux styles d'interaction dans une même architecture.

CONCLUSION

Nous avons décrit dans cet article un modèle d'architecture logicielle pour la réalisation d'applications interactives : les *graphes combinés*, qui réunit les graphes de scènes pour la description de la partie graphique, et les graphes d'interaction pour la description des interactions. Cette association conjugue donc capacités d'expression et possibilités graphiques, avec une description fine et dynamique de l'interaction. De plus, l'aspect modulaire de cette architecture se prête à l'utilisation d'outils de prototypage visuels interactifs et dynamiques, comme nous l'avons montré dans [14]. Son implémentation actuelle (http://www.emn.fr/x-info/magglite) a permis de prototyper de nombreuses interactions avancées, ainsi qu'une interface innovante de dessin pour la modélisation 3D [15]. Une future version, utilisant le moteur graphique SVG Batik [21], est en cours de développement.

BIBLIOGRAPHIE

1. Bastide, R. et Palanque, P. A Visual and Formal Glue Between Application and Interaction. *Journal of Visual Language and Computing,* 10(3). 1999.
2. Beaudouin-Lafon, M. et Lassen, H. The architecture and implementation of CPN2000, a post-WIMP graphical application. *Proc. of the 13th Annual Symposium on User Interface Software and Technology (UIST-00)*, pp 181-190, 2000. ACM Press.
3. Beaudouin-Lafon, M. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. *Proc. of the 2000 Conference on Human Factors in Computing Systems (CHI-00)*, pp 446-453, 2000. ACM Press.
4. Beaudoux, O. DoPIdom : Une boîte à outils pour la conception d'interfaces centres sur les documents XML. *Actes de la 16ème conférence francophone sur l'Interaction Homme-Machine (IHM 2004)*, pp 187-190, 2004. ACM Press.
5. Bederson, B., Hollan, J., Druin, A., Stewart, J., Rogers, D. et Proft, D. Local Tools: An Alternative to Tool Palettes. *Proc. of the 9th Annual Symposium on User Interface Software and Technology (UIST 96)*, pp 169-170, 1996. ACM Press.
6. Bederson B., Grosjean, J. et Meyer, J. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535-546, 2004.
7. Blanch, R., Beaudoin-Lafon, M., Conversy, S., Jestin, Y., Baudel, T. et Zhao, Y.P. INDIGO: une architecture pour la conception d'applications graphiques interactives distribuées. *Actes de la 17ème conférence francophone sur l'Interaction Homme-Machine (IHM'05)*, pp 139-146, 2005. ACM Press.
8. Borning, A. Thinglab - A Constraint-Oriented Simulation Laboratory. *Thèse de Doctorat*, Stanford University, Juillet 1979.
9. Chatty, S., Sire, S., Vinot, J.-L., Lecoanet, P., Lemort, A. et Mertz, C. Revisiting visual interface programming: creating gui tools for designers and programmers. *Proc. of the 17th ACM Symposium on User Interface and Software Technology (UIST 2004)*, pp 267-276, 2004. ACM Press.
10. Coutaz, J. PAC, an object-oriented model for dialog design. *Proc. INTERACT'87: 2nd IFIP International Conference on Human-Computer Interaction*, pp. 431-436, 1987.
11. Dragicevic P., et Fekete, J.-D. Étude d'une boîte à outils multi-dispositifs. *Actes de la 11ème conférence francophone sur l'Interaction Homme-Machine (IHM'99)*, pp. 33-36, 1999.
12. Dragicevic, P. et Fekete, J.-D. Input Device Selection and Interaction Configuration with ICON. *Proc. of IHM-HCI 2001*, pp. 543-448, 2001. Springer Verlag.
13. Esteban, O., Chatty, S. et Palanque, P. Whizz'ed : a visual environment for building highly interactive software. *Proc. of INTERACT'95: 5th IFIP International Conference on Human-Computer Interaction*, pp. 121-126, 1995. IOS Press.
14. Huot, S., Dumas, C., Dragicevic, P., Fekete, J.-D. et Hégron, G. The MAGGLITE post-WIMP toolkit: Draw it, connect it and run it. *Proc. of the 17th ACM Symposium on User Interface and Software Technology (UIST 2004)*, pp 257-266, 2004. ACM Press.
15. Huot, S., Dumas, C. et Hégron, G. Svalabard: Une table à dessin virtuelle pour la modélisation 3D. *Actes de la 16ème conférence francophone sur l'Interaction Homme-Machine (IHM 2004)*, pp 85-92, 2004. ACM Press.
16. Jacob, R. J. K., Deligiannidis, L., et Morrison, S. A software model and specification language for non-WIMP user interfaces. *ACM Transactions on Computer-Human Interaction*, 6(1):1-46, 1999.
17. Krasner, G. et Pope, S. A Description of the Model-View-Controller User Interface Programming. *Journal of Object Oriented Programming*, 1(3):26-49, 1988.
18. Lecolinet, E. A molecular architecture for creating advanced interfaces. *CHI Letters*, pp 135-144, 2003.
19. Myers, B. A., Giuse, D., Dannenberg, R. B., Vander Zanden, B., Kosbie, D., Pervin, E., Mickish, A. and Marchal, P. *Garnet: Comprehensive support for graphical highly-interactive user interfaces*. IEEE Computer, 23(11):71-85, November 1990.
20. Wernecke, J. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor,* 1993. Addison-Wesley Longman Publishing.
21. Page web du projet Batik: http://xmlgraphics.apache.org/batik/