ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE
M.Ing.

PAR
Erick VELÁZQUEZ-GODÍNEZ

# DES TECHNIQUES D'INTERACTION BIMANUELLES POUR LA MANIPULATION DE VISUALISATIONS DE RÉSEAUX

MONTRÉAL, LE 17 JANVIER 2012

**PRÉSENTATION DU JURY**

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Michael McGuffin, directeur de mémoire
Département de génie logiciel et des TI, École de technologie supérieure

M. Christian Desrosiers, président du jury
Département de génie logiciel et des TI, École de technologie supérieure

Mme. Sylvie Ratté, membre du jury
Département de génie logiciel et des TI, École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 20 DÉCEMBRE 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

## REMERCIEMENTS

J'arrive à la fin de ma maîtrise, et voici l'occasion idéale pour remercier tous les gens qui m'entourent et qui ont rendu possible la réalisation de ce travail.

Tout d'abord, je tiens à remercier Madame Estefania Fuentes pour avoir cru en moi dès le premier moment, pour ses conseils, sa persévérance et son soutien lors de mon adaptation dans ce beau pays. Merci Estefania, pour m'avoir fait sentir que je fais partie de ta famille. Je suis particulièrement reconnaissant à son équipe de travail au BRI de cette école. Ensuite, je remercie infiniment Dr. Alejandra Rocha, qui m'a motivé à continuer mes études, pour son soutien, son temps et son amour inconditionnel. Avec une mention très spéciale, je me dirige vers mon directeur, le professeur Michael McGuffin, pour vous remercier de votre disponibilité et de toute la patience dont vous avez fait preuve à mon égard lors de l'élaboration de ce projet. J'apprécierai toujours chaque moment de discussion avec vous, merci d'avoir partagé vos connaissances avec moi. Je tiens à remercier mes belles femmes : ma mère, ma tante Lola et mes deux sœurs : quienes han aceptado mis alegrías y más de una vez mis locuras. Miles de besos para ustedes. Je profite de l'opportunité pour remercier mes nouveaux amis, la communauté des hispanophones de l'ETS, et tous mes amis québécois qui m'ont fait découvrir la joie de l'hiver avec une délicieuse poutine et du pâté chinois ! Je remercie aussi le groupe de recherche HIFIV pour sa participation dans le projet et Christophe Viau pour son soutien technique quand j'en ai eu besoin. De la même façon, je tiens à remercier Emmanuel Pietriga, pour avoir fourni les données d'aéroports montrées dans la Figure 2.12. Je mentionne aussi le temps précieux et disponibilité de tous les participants qui nous ont donné leurs réactions face à notre prototype logiciel.

Finalement, je remercie le CONACYT pour avoir eu confiance en moi et pour m'avoir permis de réaliser un cycle supérieur à l'étranger.

# DES TECHNIQUES D'INTERACTION BIMANUELLES POUR LA MANIPULATION DE VISUALISATIONS DE RÉSEAUX

Erick VELÁZQUEZ-GODÍNEZ

## RÉSUMÉ

Ce mémoire présente les résultats d'une exploration de l'espace de conception des interfaces utilisateurs à deux mains pour interagir avec une visualisation de réseaux (c.-à-d. un graphe). Un prototype logiciel réalisé en Java, démontrant des nouvelles techniques d'interaction bimanuelles, est proposé. Le prototype utilise deux dispositifs d'entrée indirects (comme deux souris ou des dispositifs semblables, qui ne sont pas sur la surface d'affichage). L'utilisation de tels dispositifs indirects permet d'éviter les problèmes associés à l'entrée multitactile directe sur la surface d'affichage, comme l'occlusion des données par les doigts. Le prototype permet des interactions habituelles, comme la manipulation directe (translation, rotation, et changement d'échelle avec deux points d'entrée) de sous-graphes. De plus, des techniques d'interaction novatrices sont réalisées dans le prototype, comme (1) l'utilisation des deux mains pour sélectionner le plus court chemin entre deux nœuds, suivie d'un glissement pour sélectionner les voisins du chemin jusqu'à une distance voulue ; et (2) l'utilisation de la main non-dominante (MND) pour contrôler le zoom pendant que la main dominante (MD) sélectionne des nœuds le long d'un chemin. Le prototype comprend aussi (3) un HotGlass, une synthèse novatrice des widgets antérieurs de ToolGlass (Bier et al., 1993) et de HotBox (Kurtenbach et al., 1999). Le HotGlass se tient dans la MND, et peut être cliqué-à-travers par la MD comme un ToolGlass, mais peut aussi être "téléporté" sous le curseur de la MD comme s'il s'agissait d'un HotBox. Enfin, le prototype comprend (4) un autre widget novateur bimanuel, la lentille MultiVisu, qui permet de modifier la représentation visuelle d'un sous-graphe. Chaque lentille MultiVisu peut montrer une partie du réseau soit sous forme nœuds-liens, ou bien sous forme de matrice d'adjacence, ou bien sous forme de nuage de points. Plusieurs lentilles MultiVisu peuvent être affichées simultanément, permettant de configurer une visualisation hybride du graphe qui étend le genre de visualisation hybride possible avec NodeTrix (Henry et al., 2007). Le mémoire se termine par la présentation des premières réactions d'utilisateurs.

**Mot-clés:** Visualisation de réseaux, menus contextuels, interaction bimanuelle, ToolGlass, HotBox

# BIMANUAL INTERACTION TECHNIQUES FOR MANIPULATING NETWORK VISUALIZATIONS

Erick VELÁZQUEZ-GODÍNEZ

## ABSTRACT

This thesis reports the results of an exploration of the design space of two-handed user interfaces for interacting with a network visualization (i.e., a graph visualization). A software prototype implemented in Java demonstrates novel bimanual interaction techniques. The prototype uses two indirect input devices (such as two mice or similar devices, that are not on the display surface). The use of such indirect devices allows us to avoid problems associated with direct multitouch input on the display surface, such as occlusion of data by fingers. The prototype supports status quo interaction techniques, such as direct manipulation (translation, rotation, and scaling via two input points) of subgraphs. Additionally, the prototype supports novel interaction techniques, including (1) using both hands to select the shortest path between two nodes, and then dragging to select neighbors of the path out to a given distance, and (2) using the non-dominant hand (NDH) to control zoom while the dominant hand (DH) selects nodes along a path. The prototype also supports (3) a HotGlass, a novel synthesis of the pre-existing ToolGlass (Bier et al., 1993) and HotBox (Kurtenbach et al., 1999) widgets. The HotGlass is held in the NDH and can be clicked through by the DH like a ToolGlass, but can also be "teleported" underneath the DH's cursor as if it were a HotBox. Finally, the prototype also demonstrates (4) another novel bimanual popup widget called the MultiVis Lens, that can transiently modify the visual representation of a subgraph. Each MultiVis Lens displays a local portion of the network in either node-link, adjacency matrix, or scatterplot form. Several MultiVis Lenses can be popped up and left on-screen simultaneously, like tear-off menus, enabling a hybrid network visualization to be assembled that extends the kind of hybrid visualization possible with NodeTrix (Henry et al., 2007). At the end of this thesis, initial user feedback is reported.

**Keywords:** Network visualization, popup widgets, bimanual interaction, ToolGlass, HotBox

## TABLE DES MATIÈRES

Page

## LISTE DES FIGURES

Page

## LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

| | |
|---|---|
| ETS | École de Technologie Supérieure |
| GUI | *Graphical User Interface* (Interface Graphique Utilisateur) |
| MD | Main Dominante |
| MND | Main Non-Dominante |
| NodeTrix | (visualisation hybride mélangeant diagrammes *node-link* et *matrix*) |
| SPLOM | *ScatterPLOt Matrix* (matrice de nuages de points) |
| tâche MC | tâche Multi-Commande |
| tâche MO | tâche Multi-Objet |
| tâche MCMO | tâche Multi-Commande et Multi-Objet |

_

# INTRODUCTION

Les graphes (c.-à-d. réseaux) sont très utiles pour modéliser beaucoup de situations, comme les réseaux sociaux, l'internet et le web, les réseaux de téléphones cellulaires, les réseaux biologiques, les routes entre des villes, et les composants d'un logiciel. Depuis plusieurs années, des utilisateurs et des chercheurs dans plusieurs domaines rencontrent des graphes de plus en plus gros, et ont besoin de visualiser ces graphes. Heureusement, le domaine du *graph drawing* identifie, depuis des années, différents algorithmes pour calculer le *layout* (disposition) d'un graphe (Di Battista *et al.*, 1999; Kaufmann et Wagner, 2001), permettant la génération automatique d'une visualisation. Toutefois, il existe encore un besoin de pouvoir interagir avec cette visualisation. Par exemple, les utilisateurs peuvent avoir besoin de personaliser la disposition ou les propriétés de certains nœuds ou sous-graphes, peut-être pour mettre en évidence certaines informations selon le contexte, ou peut-être parce que le nombre de nœuds est tellement grand que l'utilisateur ne peut comprendre qu'une visualisation d'un sous-graphe à la fois. Donc, une interface utilisateur permettant cela reste nécessaire.

Historiquement, les GUIs (*Graphical User Interfaces* ou interfaces utilisateurs graphiques) ont changé le paradigme d'interfaces où les utilisateurs saisissent des commandes textuelles pour réaliser des tâches. En se servant d'objets visuels représentatifs, les GUIs permettent de faire les mêmes tâches dans un environnement graphique via la manipulation directe des objets qui y sont contenus. Cette façon d'interagir peut réduire le temps d'apprentissage pour une interface, car l'utilisateur peut *voir* les affordances (Norman, 2002) (commandes et objets) qui lui sont disponibles, au lieu d'avoir à mémoriser des commandes textuelles. Les GUIs peuvent aussi augmenter la vitesse de réalisation du travail et le niveau de satisfaction des utilisateurs (Shneiderman, 1983). Bien sûr, les GUIs sont aussi le paradigme d'interface naturel pour des visualisations d'information, et il existe maintenant plusieurs logiciels graphiques pour la visualisation de graphes, comme Tulip [1], Graphviz [2], Gephi [3], Pajek [4], et Walrus [5]. Toutefois, il

1. http://www.tulip-software.org/
2. http://www.graphviz.org/
3. http://gephi.org/
4. http://pajek.imfm.si/
5. http://www.caida.org/tools/visualization/walrus/

existe au moins deux problèmes généraux avec les GUIs : premièrement, comment permettre un accès rapide et fréquent aux commandes, surtout lorsqu'il y a un grand nombre de commandes possibles ; et deuxièmement, comment permettre à l'utilisateur de s'exprimer et d'interagir rapidement, surtout vu que l'utilisateur est souvent limité à un simple clavier et souris.

Pour résoudre le premier problème, plusieurs widgets contextuels (*popup widgets*) ont été proposés, qui peuvent être affichés au besoin, et qui contiennent des commandes exécutables. Comme ces widgets peuvent être activés à la position actuelle du curseur, ils évitent d'avoir à faire des mouvements aller-retour entre l'espace de travail et des panneaux de boutons ou des barres d'outils sur la périphérie de l'écran. Ils ont aussi l'avantage de seulement consommer de l'espace à l'écran quand l'utilisateur les affiche, permettant de consacrer plus d'espace aux données en dessous. De plus, certains widgets contextuels permettent une interaction gestuelle (Callahan *et al.*, 1988; Kurtenbach et Buxton, 1993; Pook *et al.*, 2000; Guimbretière et Winograd, 2000; McGuffin *et al.*, 2002) qui est plus rapide que la sélection de commandes dans un menu contextuel linéaire habituel (Figure 1.3). (Ces widgets sont discutés dans la section 1.2 de ce mémoire.)

Pour attaquer le deuxième problème, une solution est de permettre plus de points d'entrée, en captant la position de plusieurs dispositifs ou plusieurs doigts en même temps. L'entrée multitactile, avec plusieurs doigts directement sur la surface de l'écran, est une forme d'entrée qui jouit de beaucoup d'intérêt des chercheurs et du public dans les dernières années, et permet plus de degrés de liberté qu'une simple souris. Toutefois, l'entrée multitactile souffre de problèmes, comme le fait que les doigts des utilisateurs sont relativement gros, créant de l'occlusion et rendant difficile la sélection de points précis. Habituellement, l'entrée multitactile ne permet pas non plus de distinguer entre les états de survol et glissement (Figure 1.2), contrairement à une souris. L'entrée multitactile ne permet donc pas de réaliser des infobulles (*tooltips*), ou d'autres retours qui dépendent d'un survol, avant un pressement de bouton. Une approche alternative, qui évite ces problèmes, est d'utiliser deux dispositifs de pointage indirects (c.-à-d. pas sur l'écran), comme deux souris. Il s'agit d'entrée bimanuelle (Kabbash *et al.*, 1994), et augmente le nombre de degrés de liberté à au moins 4 (parfois même 6, si les dispositifs captent

l'orientation des mains) tout en évitant les problèmes d'occlusion et de précision des doigts, et en permettant l'opération de multiples boutons sur chaque dispositif, permettant de distinguer entre les états de survol et de glissement. Le ToolGlass (Bier *et al.*, 1993; Kurtenbach *et al.*, 1997) est une technique d'interaction bimanuelle permettant de sélectionner une commande et un objet en même temps, en cliquant-à-travers une palette d'outils [6] qui se tient dans la main non-dominante (MND). Le HotBox (Kurtenbach *et al.*, 1999) est une autre sorte de technique d'interaction, quasi-bimanuelle, qui utilise une souris dans la main dominante (MD) et un clavier dans la MND simultanément [7]. (Les travaux antérieurs en interaction bimanuelle sont discutés dans la section 1.3 du mémoire.)

Toutes les solutions mentionnées peuvent être appliquées à l'interaction avec les visualisations des graphes, et plusieurs l'ont déjà été (interaction multitactile avec les graphes (Frisch *et al.*, 2009; Dwyer *et al.*, 2009; Schmidt *et al.*, 2010) ; ToolGlass pour les graphes (Beaudouin-Lafon *et al.*, 2001) ; HotBox pour un graphe (McGuffin et Jurisica, 2009)). Malgré ce travail antérieur, ce mémoire montrera qu'il reste encore des opportunités pour innover et inventer des nouvelles techniques d'interaction bimanuelles pour les visualisations de graphes, permettant une interaction plus rapide qu'il ne serait possible avec une seule souris, et permettant d'accéder à un grand nombre de commandes. Dans la section 2.1 (et la Figure 2.1), on identifie différentes sortes d'interactions bimanuelles possibles pour les tâches associées aux graphes. Ensuite, dans le chapitre 2, on présente quatre nouvelles techniques d'interaction bimanuelles pour les visualisations de graphes : (1) le HotGlass, un widget qui fait une synthèse de ToolGlass et HotBox ; (2) une technique bimanuelle pour sélectionner le plus court chemin, et les voisins du chemin, entre deux nœuds ; (3) une technique bimanuelle pour parcourir un chemin de nœuds, en utilisant la MND pour zoomer et la MD pour sélectionner des nœuds ; et (4) une lentille bimanuelle permettant de changer la représentation visuelle de sous-graphes. Un prototype réalisé en Java démontre ces quatre techniques appliquées à la visualisation de graphes. Enfin, des premières réactions d'utilisateurs sont présentées.

6. Voir `http://www.youtube.com/watch?v=fUwYCbhFj1U` pour une démonstration.
7. Voir `http://www.youtube.com/watch?v=U2vhjdnSfsQ` pour une démonstration.

# CHAPITRE 1

# REVUE DE LITTÉRATURE

Ce chapitre fait un survol de sujets reliés aux interfaces graphiques bimanuelles et la visualisation interactive de graphes. D'abord, nous abordons les côtés matériel (dispositifs de pointage, section 1.1) et logiciel (widgets, section 1.2) des interfaces graphiques. Ensuite, nous examinons la manipulation bimanuelle (section 1.3), encore aux niveaux matériel et logiciel. Finalement, nous regardons la visualisation des graphes (section 1.4) pour arriver aux des questions de recherche (sections 1.5-1.6) qui seront investiguées dans le prochain chapitre.

## 1.1 Les dispositifs de pointage

Les dispositifs d'entrée sont des périphériques (pièces de matériel) utilisés pour fournir des données et signaux de contrôle au système de traitement de l'information (un ordinateur). Les dispositifs d'entrée peuvent être classés selon la modalité d'entrée (mouvement mécanique, audio, visuel, etc.) et aussi selon si l'entrée est discrète (p. ex. appuie sur une touche) ou continue (p. ex. la position d'une souris). Parmi les dispositifs d'entrée, on retrouve plusieurs sortes de claviers, de boutons, les microphones, les appareils vidéo, et les dispositifs de pointage.

Dans le contexte des interfaces graphiques, nous nous intéressons particulièrement aux dispositifs de pointage, qui permettent de spécifier une position, une orientation, ou une valeur sur un ou plusieurs axes. À titre d'exemples : les souris, boules de commande (*trackballs*), écrans tactiles (*touchscreens*), tablettes graphiques, pavés tactiles (*touchpads*), et les manettes (*joysticks*).

On peut classer les dispositifs de pointage selon

- Le nombre de degrés de liberté impliqués. Par exemple, les souris traditionnelles ont deux degrés de liberté, tandis que certains dispositifs pour les logiciels 3D peuvent avoir 3 ou même 6 degrés de liberté (3 pour la position, et 3 pour l'orientation).

- Si l'entrée est *directe* ou *indirecte*. Avec l'entrée directe, l'espace d'entrée coïncide avec l'espace d'affichage, c.-à-d. que le pointage se fait où la rétroaction visuelle ou le curseur apparaît. Les écrans tactiles et les stylos à lumière impliquent l'entrée directe. Par contre, les souris et les boules de commande impliquent une entrée indirecte.
- Si l'information de position est absolue (p. ex. sur un écran tactile) ou relative (p. ex. avec une souris qui peut être soulevée et repositionnée).

Buxton (1983) propose une taxonomie des dispositifs de pointage selon 4 dimensions, tel le nombre de degrés de liberté (Figure 1.1). Une telle taxonomie sert à résumer plusieurs comparaisons, et peut aussi servir à mettre en évidence les "cases vides" qui pourrait inspirer l'invention de combinaisons novatrices de propriétés chez un nouveau dispositif.

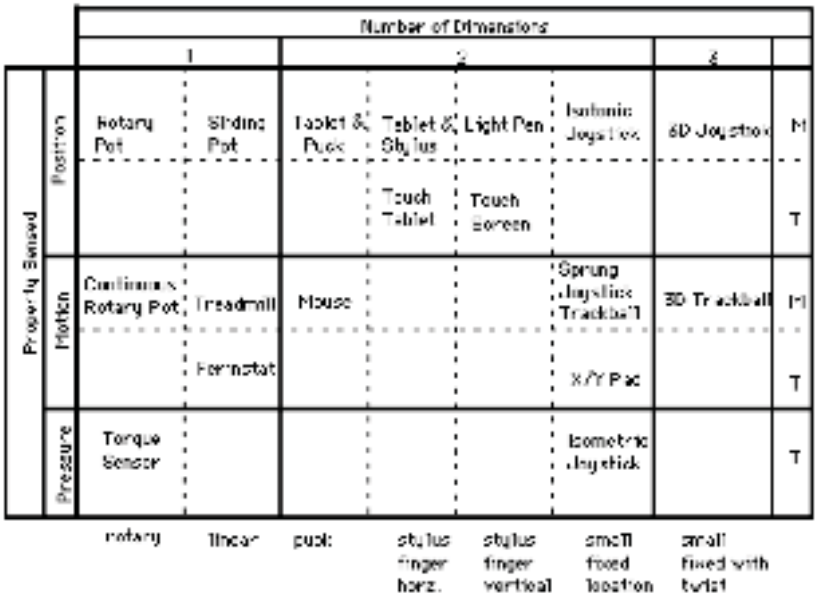


Figure 1.1 Une taxonomie de dispositifs de pointage, tiré de Buxton (1983). Les rangées correspondent à la propriété captée, et les sous-rangées correspondent aux dispositifs utilisant soit le toucher direct ("T") ou bien un intermédiaire mécanique ("M") comme un stylet. Les colonnes correspondent au nombre de degrés de liberté, et les sous-colonnes correspondent aux différents mouvements de main et de bras.

Plus tard, Buxton (1990) présenta un modèle à trois états des dispositifs de pointage (Figure 1.2). Un dispositif dans l'état 0, "hors de portée", sait que la main (ou le dispositif) n'est pas dans une position prête à fournir des informations. Par contre, les états 1 et 2, "survol" et "glissement", fournissent une position $(x, y)$, par exemple. Une souris ne peut détecter l'état 0 (quand la main n'est pas sur la souris), mais peut toutefois distinguer entre les états 1 et 2, même si la souris a seulement un bouton. La souris fournit donc une position $(x, y)$ dans ses deux états. Par contre, un écran tactile détecte les états 0 et 1, mais n'a pas d'état 2. L'écran tactile, donc, fournit une position $(x, y)$ dans seulement un de ses deux états, et ne permet donc pas des effets de survol (comme les infobulles). De ce point de vue, la souris est plus expressive que l'écran tactile, même si la souris a seulement un bouton. Comme troisième exemple, les tablettes graphiques avec stylet sont souvent capables de détecter les trois états : un stylet hors de portée, un stylet juste au dessus de la tablette, et un stylet en contact avec la tablette. La tablette graphique est donc encore plus expressive que la souris ou l'écran tactile.

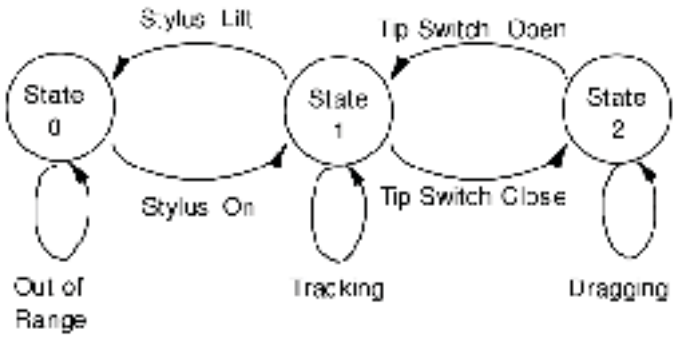


Figure 1.2 Le modèle à trois états des dispositifs de pointage, tiré de Buxton (1990). "Out of Range", "Tracking", et "Dragging" correspondent aux états "hors de portée", "survol", et "glissement". Remarquez que les états de survol et glissement fournissent une position $(x, y)$, contrairement à l'état hors de portée.

### 1.1.1 La souris

Une souris est un dispositif de pointage qui fonctionne en détectant les mouvements relatifs à deux dimensions de la surface où elle se trouve. Physiquement parlant, une souris consiste en un objet tenu dans l'une des mains de l'utilisateur avec un ou plusieurs boutons.

Balakrishnan *et al.* (1997) énumèrent les caractéristiques qui ont permis à la souris d'être acceptée comme dispositif de pointage depuis son apparition :

a. La forme : La forme physique de la souris, associée au fait qu'elle opère sur une surface horizontale, assure que l'utilisateur ne soit limité par aucune adhérence. En plus, le bras de l'utilisateur repose sur une table ou un bureau lors du mouvement, réduisant ainsi la fatigue.

b. La stabilité : Étant donné que la souris est légère et qu'elle a une grande zone en contact avec la surface lorsque déplacée, le tremblement de la main de l'utilisateur est réduit, permettant une opération avec précision. Également, la souris est souvent dans un état stable permettant son utilisation sans devoir être déplacée.

c. Le mode relatif : Les dispositifs d'entrée peuvent soit reporter leur position mesurée absolue ou leur position actuelle relative à un point clé. Comme la souris est un dispositif relatif avec embrayage implicite, la quantité du mouvement de la main requise pour l'utiliser peut être très petite. Donc, l'utilisateur n'a pas besoin de développer beaucoup d'effort quand il travaille avec la souris.

d. Le mappage du curseur : Le mappage par défaut du mouvement de la souris est naturel, c'est-à-dire que lorsque la souris se déplace vers le haut, le curseur le fait aussi.

e. La position des boutons : La direction du mouvement de la souris est orthogonale aux dimensions sensées par les boutons. Il devient alors facile d'opérer les boutons sans bouger par inadvertance le curseur.

Balakrishnan *et al.* (1997) mentionnent aussi que les caractéristiques citées précédemment devraient être considérées pour la conception de tout nouveau dispositif d'entrée.

## 1.2 Les widgets contextuels (*popup widgets*)

### 1.2.1 Sélection de commandes et d'objets

Dans une grande fraction (peut-être la majorité) des interfaces graphiques, les actions prises par l'utilisateur se réduisent à des sélections de commandes (ou opérations, ou verbes) et d'objets (ou opérandes, ou noms) sur lesquels les commandes seront appliquées. Prenons une application de dessin comme exemple. Souvent, plusieurs commandes se trouveront dans une barre d'outils (*toolbar*) ou une palette flottante d'icônes, et l'utilisateur aura à cliquer sur une commande pour entrer dans un mode associé (par exemple, un mode crayon, ou un mode efface). Une fois dans le mode, l'utilisateur pourra appliquer la commande à des positions ou sur des objets situés sur une toile virtuelle. Ce paradigme d'interaction, avec une barre d'outils et des modes, se trouve dans plusieurs logiciels, mais entraîne certains problèmes. D'abord, les barres d'outils et palettes de widgets occupent de l'espace à l'śsgsfecran, laissant moins pour les documents (dessins, données) qui sont l'objet d'intérêt principal de l'utilisateur. Deuxièmement, l'utilisateur peut se trouver à faire beaucoup de mouvements aller-retour pour passer des barres d'outils dans la périphérie à la toile ou l'espace de travail. Troisièmement, lorsqu'il y a plusieurs modes possibles dans une interface, il y a la possibilité que l'utilisateur soit porté à faire des erreurs de mode, c'est-à-dire essayer d'appliquer un outil quelconque en pensant qu'il est actuellement dans le mode d'un autre outil. (Sellen *et al.*, 1992).

Les widgets contextuels, par exemple les menus contextuels, offrent une façon prometteuse d'éviter ces problèmes. Ces widgets sont affichés sur demande, là où le curseur est déjà situé, et ne nécessitent donc pas de mouvements aller-retour, et occupent seulement de l'espace à l'écran lorsque l'utilisateur les fait afficher. De plus, on peut concevoir les widgets contextuels de manière à éviter les modes de commande, et donc éviter les erreurs de mode. L'utilisateur peut d'abord placer son curseur sur l'objet (ou la position) où il veut appliquer une commande, ensuite cliquer pour afficher le widget contextuel, sélectionner la commande, et relâcher pour

exécuter la commande sur l'objet. De cette manière, l'utilisateur peut toujours être dans un mode de sélection d'objets par défaut, et entrer en mode commande seulement de manière transitoire lorsque le widget est affiché. De plus, ce genre d'interaction fusionne la sélection d'objet et de commande en un seul "geste" de pressement, glissement, et relâchement (Buxton, 1986).

### 1.2.2 Les menus contextuels linéaires

Le widget contextuel le plus connu et courant est le menu contextuel linéaire (Figure 1.3). En anglais, on parle de *linear popup menu*, le mot *popup* mettant en évidence que le menu est affiché par-dessus l'interface principale, de façon transitoire. Le terme français "menu contextuel" fait référence au fait que, souvent, les commandes disponibles dans le menu seront fonction de l'objet en-dessous du menu ou de la fenêtre ou autre contexte.

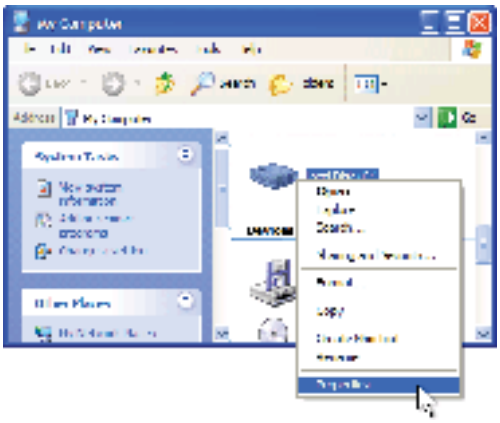

Figure 1.3 Menu contextuel linéaire, activé avec le bouton droit de la souris dans Microsoft Windows.

### 1.2.3 Les menus circulaires

Le menu circulaire (Wikipedia, 2011) (Figure 1.4), aussi connu comme menu radial (ou *radial menu* ou *pie menu* en anglais) est un menu contextuel avec les items disposés autour du curseur. De cette manière, la distance à faire avec le curseur est la même pour chaque item. Contraire-

ment au menu linéaire, où les items plus loin prennent plus de temps à aller chercher, les items dans un menu circulaire impliquent tous un temps équivalent. De plus, si le menu circulaire est bien conçu, il permet à l'utilisateur de glisser et relâcher simplement dans la direction d'un item, sans nécessairement relâcher par dessus l'étiquette de l'item. Cela permet à l'utilisateur de sélectionner chaque item avec une direction de glissement différente, sans se soucier de la longueur du glissement, permettant donc des glissements rapides et approximatifs. Le résultat net est que le menu circulaire permet un gain de vitesse d'environ 15-20% selon Callahan *et al.* (1988). En plus de ce gain en vitesse, le menu circulaire conserve tous les avantages des widgets contextuels mentionnés dans la section précédente. Par contre, un désavantage du menu circulaire est qu'il limite le nombre d'items à environ 8 items par niveau du menu, car un nombre excessif d'items engendre un taux d'erreur élevé (Kurtenbach et Buxton, 1993).

Figure 1.4 Un menu circulaire, une sorte de menu contextuel, tiré de Wikipedia (2011).

Les menus circulaires fonctionnent bien avec soit une souris ou un stylet comme dispositif d'entrée. Le "trou" au milieu du menu donne un moyen de quitter le menu. Si le menu circulaire doit contenir des sous-menus, alors une tranche du menu principal peut conduire à un autre menu circulaire. En sélectionnant cette tranche, le sous-menu qui s'ouvre sera centré sur le curseur.

### 1.2.4 Les Marking Menus

Un Marking Menu (Kurtenbach et Buxton, 1993, 1994) est une sorte de menu circulaire. Comme le menu circulaire normal, le Marking Menu permet une hiérarchie de sous-menus (Figure 1.5, A). De plus, le Marking Menu permet de réaliser une sélection selon deux modes différents : soit en affichant le menu, ou bien en faisant un geste en direction de l'item désiré sans faire apparaître le menu. L'interaction peut se passer de deux manières différentes : (1) l'utilisateur appuie un bouton du dispositif et attend environ 0.3 secondes pour faire apparaître le menu, qu'il peut ensuite naviguer pour sélectionner l'item désiré, ou bien (2) l'utilisateur appuie un bouton et commence immédiatement à tracer un geste en direction de l'item désiré (Figure 1.5, B et C). Dans le deuxième cas, le menu n'apparaît pas. Un nouvel utilisateur va normalement utiliser la première méthode, mais avec le temps, en sélectionnant les mêmes items à plusieurs reprises, l'utilisateur apprendra les gestes correspondants, et pourra commencer à exécuter ces gestes plus rapidement et passer donc à la deuxième méthode d'interaction.

Les Marking Menus étendent donc les menus circulaires en permettant d'accéder rapidement aux items avec des traits polygonaux qui sont interprétés en utilisant la reconnaissance des gestes. Tel comme le montre la Figure 1.5, la reconnaisance de ces gestes est invariante avec l'échelle, alors un geste réalisé en petit ou gros sera interprété de la même façon. Notons que selon une étude de Kurtenbach et Buxton (1993), il est conseillé de ne pas dépasser environ 8 items par sous-menu, et 3 niveaux de profondeur, dans un Marking Menu. Sinon, le taux d'erreur de l'utilisateur qui fait des gestes augmente. Un seul Marking Menu peut alors permettre d'accéder à environ $8 \times 8 \times 8$ items, ou quelques centaines d'items, avec des gestes.

Remarquez aussi que chez les menus traditionnels qui offrent des raccourcis clavier aux items, l'utilisateur doit faire un effort conscient pour remarquer et apprendre les raccourcis clavier avant de pouvoir les exploiter. Par contre, dans un Marking Menu, l'apprentissage des gestes se fait tout naturellement, car l'utilisateur qui navigue le menu exécute les mêmes mouvements

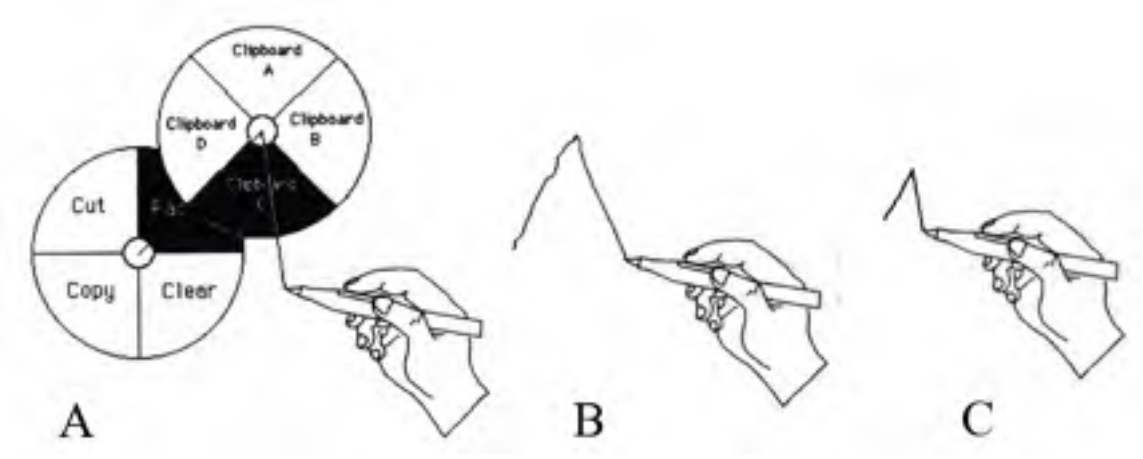


Figure 1.5 A : Un Marking Menu, une sorte de menu circulaire, ici contenant un sous-menu. B : Au lieu de naviguer le menu et le sous-menu, l'utilisateur peut effectuer un geste rapide pour sélectionner le même item qu'en A. Le Marking Menu interprète la forme du geste. C : Le même geste en plus petit sélectionnera aussi le même item, car la reconnaissance de geste fait par le Marking Menu ne varie pas avec l'échelle. Cela permet de faire des gestes petits et rapides. Adapté de Kurtenbach (1993).

(en plus lent) qui correspondent au geste accéléré. Les Marking Menus ont donc la propriété de permettre un passage graduel et naturel vers l'usage expert.

### 1.3 La manipulation bimanuelle

Avec la manipulation bimanuelle, l'utilisateur interagit avec le système avec un dispositif d'entrée dans chacune de ses mains, par exemple, avec deux souris, ou bien les deux mains sur une surface multitactile. Elle porte un intérêt spécial par ses nombreux avantages :

- Le temps nécessaire pour effectuer des tâches peut être réduit, pour deux raisons distinctes. D'abord, si les deux mains sont capables de travailler en parallèle, elles pourront exêcuter deux tâches en même temps (par exemple, la translation de deux objets simultanément) ou effectuer une tâche complexe à plusieurs degrés de liberté (par exemple, la translation, rotation, et changement d'échelle d'un rectangle en même temps) qui aurait été décomposée en plusieurs tâches simples dans une interface unimanuelle. Deuxièmement, le temps peut être réduit parce que l'utilisateur n'a pas besoin de passer autant de temps à *changer* de modes : par exemple, au lieu de basculer entre des modes de trans-

lation, rotation, et changement d'échelle, un mode peut suffire pour ces trois opérations (Balakrishnan et Kurtenbach, 1999; Owen *et al.*, 2005).

- Un nombre réduit de modes dans les interfaces bimanuelles implique que ces interfaces pourront être plus faciles à comprendre et porter à moins d'erreurs. Par exemple, la main gauche pourrait toujours servir à zoomer, tandis que la main droite sert à faire des sélections. Cela permet d'éviter des erreurs de mode (Sellen *et al.*, 1992).
- L'exploitation des habilités existantes des utilisateurs peut rendre les interfaces bimanuelles plus faciles à apprendre. La vie quotidienne implique plusieurs tâches bimanuelles connues par les utilisateurs, comme la conduite d'une voiture où la main gauche tient le volant pendant que la main droite change de vitesse, et l'écriture sur papier où une main tient le papier en place et l'autre écrit. Ces tâches peuvent servir de métaphores pour aider à apprendre des nouvelles interactions bimanuelles sur les ordinateurs (Balakrishnan et Kurtenbach, 1999; Owen *et al.*, 2005).
- L'expressivité : Owen *et al.* (2005) définit l'expressivité d'une technique d'interaction comme sa capacité de permettre à l'utilisateur de rapidement explorer des solutions et fureter ses données. Owen *et al.* (2005) présentent des résultats qui pourraient être interprétés comme une démonstration que l'interaction bimanuelle a une expressivité augmentée, permettant une meilleure performance.

Si l'on considère qu'un dispositif d'entrée fonctionne comme une extension du corps humain dans les systèmes informatiques, alors les techniques bimanuelles transfèrent l'usage habituel des deux mains que les êtres humains utilisent pour la réalisation des tâches quotidiennes.

Pour abstraire les différences entre les gens droitiers et gauchers, nous utiliserons les termes Main Dominante (MD) et Main Non-Dominante (MND) pour faire référence aux deux mains. Une interface bimanuelle peut être soit symétrique (les deux mains jouent des rôles équivalents) ou bien asymétrique (les deux mains font des choses différentes).

### 1.3.1 Choix matériels pour la manipulation bimanuelle

Souvent, les interfaces bimanuelles comprennent un dispositif de pointage 2D dans chaque main, comme deux souris, une souris et un stylet, ou les deux doigts index sur un écran multitactile. Parfois, les dispositifs permettent de détecter la position 3D des mains ou des doigts, et/ou l'orientation, et/ou permettent plusieurs doigts avec des positions indépendantes.

Parmi les nombreuses possibilités de matériel, deux qui sont devenues courantes sont (1) deux souris et (2) un écran multitactile. Un écran multitactile offre les avantages d'être direct (c.-à-d. que les mains travaillent directement sur l'espace de retour visuel, ce qui peut être plus intuitif pour un utilisateur) et de permettre, à la limite, 10 doigts $\times$ 2 degrés de liberté par doigt = 20 degrés de liberté.

L'utilisation de deux souris, par contre, offre les avantages d'éviter une occlusion du retour visuel par les doigts, de permettre la sélection de positions précises (contrairement aux doigts), et de distinguer entre les états de survol et glissement (Figure 1.2), permettant alors des effets comme les infobulles. De plus, si les souris sont munies de plusieurs boutons, l'ordinateur peut facilement différencier le fait qu'un bouton soit pressé plutôt qu'un autre, contrairement à une surface multitactile ou les doigts sont difficilement distingués. Étant donné ces avantages, le prototype présenté dans le prochain chapitre utilise deux dispositifs de pointage indirect, comme deux souris, pour effectuer l'entrée bimanuelle.

### 1.3.2 Manipulation bimanuelle symétrique

#### 1.3.2.1 Manipulation directe bimanuelle

La technique bimanuelle la plus connue est la manipulation directe avec deux points d'entrée (Figure 1.6) (Hancock *et al.*, 2006). L'utilisateur a 4 degrés de liberté, ce qui est assez pour spécifier une translation, rotation, et changement d'échelle uniforme complet. Ce genre d'interaction est souvent vu dans les interfaces multitactiles récentes, et il est même possible avec une seule main, en utilisant deux doigts en un geste de "pincement", comme sur le iPhone de

Apple pour zoomer sur une photo. Toutefois, cette manipulation directe bimanuelle était déjà possible il y a 15 ans (Kurtenbach *et al.*, 1997).


Figure 1.6 La manipulation directe, avec deux points d'entrée, permet de faire une translation, rotation, et changement d'échelle simultanément. Les deux points d'entrée peuvent être fournis par deux doigts, deux souris, ou deux autres dispositifs de pointage.

Tout comme cette manipulation sert à repositionner et redimensionner un seul objet, elle sert aussi à faire une translation et un zoom de caméra 2D dans un monde virtuel 2D, tel que permis dans le système de Kurtenbach *et al.* (1997). Par exemple, un déplacement des deux pointeurs ensemble translate la caméra, et un rapprochement ou un éloignement des deux pointeurs fait un zoom-out ou un zoom-in, respectivement.

#### 1.3.2.2 symSpline

symSpline (Latulipe *et al.*, 2006) est une technique symétrique à deux souris pour la manipulation des courbes spline, où les deux curseurs contrôlent les positions des extrémités de la tangente à un point d'édition (Figure 1.7). En bougeant la tangente avec les deux souris, la tangente et le point d'édition peuvent être déplacés, et la courbure au point d'édition est contrôlée par la longueur de la tangente. Cette technique permet à l'utilisateur d'effectuer la translation, rotation, et changement de longueur de la tangente en un seul geste symétrique et facile à apprendre. Une métaphore qui peut aider l'utilisateur à comprendre l'interaction est d'imaginer que ses deux mains tiennent un volant de bicyclette.

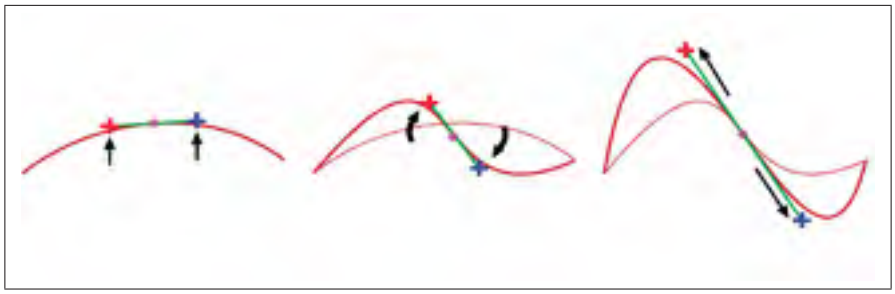

Figure 1.7 La technique d'interaction symSpline pour modifier une courbe spline. Les croix rouge et bleu indiquent les extrémités des tangentes où les curseurs agissent pour éditer la courbe. Tiré de Latulipe *et al.* (2006)

### 1.3.3 Manipulation bimanuelle asymétrique

#### 1.3.3.1 Le modèle de chaîne cinématique de Guiard

Une étude qui a beaucoup influencé les interfaces bimanuelles est celle de Guiard (1987), où il définit le rôle des mains de la façon suivante :

a. Main Dominante (MD) : Celle qui réalise le travail précis et minutieux. Elle trouve sa référence spatiale à partir de l'activité de la main non-dominante.

b. Main Non-Dominante (MND) : celle qui contribue à une action stabilisatrice pour la réalisation de la tâche globale.

Les deux mains sont considérées comme deux moteurs abstraits assemblés dans une liaison en série, formant ainsi une chaîne de coopération cinématique. À partir de cette définition Guiard tire les trois concepts suivants :

a. La MND fournit le cadre de référence par rapport à laquelle la MD réalise ses mouvements. (Par exemple, la MND tourne et ensuite tient une feuille de papier en place, ensuite la MD dessine.)

b. Les deux mains effectuent des mouvements sur des échelles spatio-temporelles asymétriques. Les mouvements de la MND sont moins fréquents et plus grossiers que ceux de la MD. La MD travaille à une échelle plus fine, en termes d'espace et de temps.

c. La précédence de la MND : la MND se déplace d'abord, pour établir le cadre de référence pour la MD, ensuite la MD se déplace.

Depuis son apparition, le modèle de la chaîne cinématique a été souvent utilisé par les chercheurs pour mieux comprendre et justifier la conception de nouvelles techniques d'interaction bimanuelles.

Le modèle de chaîne cinématique ne s'applique pas nécessairement à l'interaction bimanuelle symétrique, comme celle de la Figure 1.6. Mais, pour l'interaction *asymétrique*, le modèle prédit que des manipulations où la MND se déplace par rapport à un cadre de référence établie par la MD ne seraient pas naturelles. En effet, des interactions à deux mains qui sont mal conçues peuvent être pires qu'une interface unimanuelle (Balakrishnan et Kurtenbach, 1999; Kabbash *et al.*, 1994).

#### 1.3.3.2 Manipulation 3D asymétrique

Balakrishnan et Kurtenbach (1999) présentent un travail où la MND contrôle la rotation de la vue de caméra dans un environnement 3D, laissant la MD exécuter d'autres tâches de manipulation dans la scène 3D. La rotation de la vue de caméra par la MND se justifie par la métaphore de l'utilisateur qui tient l'objet 3D virtuel dans la main gauche (et peut alors tourner cet objet avec leur MND, pour faciliter des manipulations fines par la MD) ou bien la MND qui tourne un tour de potier pour la sculpture.

Dans leur interface, la manipulation de la caméra (avec la MND) et la sélection avec le curseur (MD) peuvent se réaliser simultanément. Les auteurs ont trouvé que le contrôle de la caméra par la MND est bénéfique pour les tâches en 3D, et qu'il augmente l'impression d'engagement avec l'interface.

Dans la section 2.3.4, nous reviendrons sur l'idée de contrôler une caméra avec la MND pendant que la MD fait d'autres manipulations.

#### 1.3.3.3 Le ToolGlass

Le ToolGlass (Bier *et al.*, 1993; Kurtenbach *et al.*, 1997) est une palette d'outils semi-transparente qui suit la MND, comme si la MND la tenait (Figure 1.8, gauche). La MD, pour sa part, déplace un curseur conventionnel. La particularité du ToolGlass est qu'il permet d'être "cliqué-à-travers". En cliquant-à-travers, l'utilisateur se trouve à sélectionner à la fois un outil (commande), et l'endroit où la commande doit être appliquée. Par exemple, si le clic-à-travers est sur un outil de création de rectangle, par-dessus une toile, alors le rectangle va apparaître à cette position, et ensuite, tant et aussi longtemps que l'utilisateur garde son bouton (de MD) enfoncé, il pourra repositionner ses mains pour placer les coins du rectangle (et pendant ce glissement, le ToolGlass peut être caché) [1]. Au moment du relâchement, la création du rectangle est finie, et le ToolGlass réapparaît. Les cliques-à-travers peuvent aussi se faire par-dessus des objets qui

1. Voir `http://www.youtube.com/watch?v=fUwYCbhFj1U` pour une démonstration.

existent déjà. Par exemple, un clic-à-travers d'un outil de couleur jaune, par dessus un objet rouge, aurait comme effet de changer la couleur de l'objet au jaune.

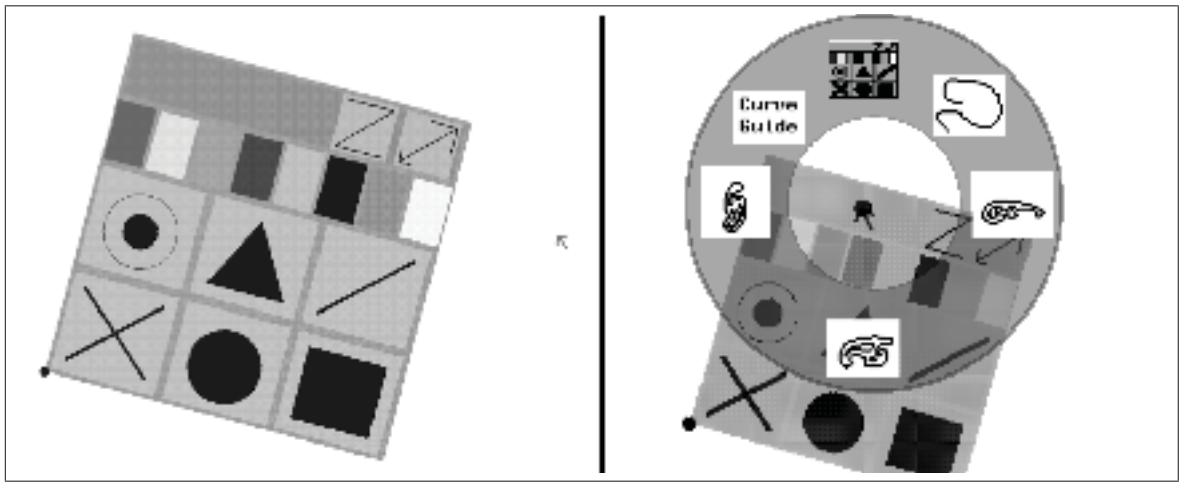


Figure 1.8 Un ToolGlass, adapté de T3 (Kurtenbach *et al.*, 1997). À gauche, le ToolGlass se tient dans la main gauche (MND), et un curseur est contrôlé par la main droite (MD). À droite, un menu circulaire est activé par la MD en cliquant sur le ToolGlass. Le menu circulaire permet de sélectionner parmi différentes versions du ToolGlass, dont certaines qui contiennent des règles perroquets pour tracer des courbes lisses.

Par rapport à la discussion de la section 1.2.1, on voit qu'avec le ToolGlass, il est possible de sélectionner une commande et un objet de façon simultanée, car les deux mains sont impliquées.

En plus des commandes "clic-à-travers", le ToolGlass peut offrir d'autres fonctionnalités. Par exemple, le ToolGlass de Kurtenbach *et al.* (1997) est muni d'un menu circulaire permettant de basculer entre différents contenus de ToolGlass (Figure 1.8, droite). L'utilisateur peut basculer vers une palette en forme de "règle perroquet" (*French curve*), permettant de contraindre les tracées de crayon de la MD. Dans les travaux de Bier *et al.* (1993), on proposa aussi des "lentilles magiques" qui suivent la MND et servent à modifier la présentation graphique des objets en dessous de la lentille, ce qui peut servir à montrer des informations cachées.

Kabbash *et al.* (1994) ont effectué une expérience pour comparer le ToolGlass avec d'autres techniques de sélection de commandes et d'objets, avec une et deux mains. Ils ont trouvé que le ToolGlass était la technique la plus rapide, en plus d'être la technique pour laquelle l'utilisateur se servait le plus de sa MND.

#### 1.3.3.4 Le HotBox

Contrairement aux autres techniques bimanuelles présentées ici, le HotBox n'implique qu'un seul dispositif de pointage (une souris), dans la MD. La MND, pour sa part, sert à activer simplement une touche de clavier. On pourrait alors dire que le HotBox est "tout juste" bimanuel, ou quasi-bimanuel. Toutefois, la touche de la MND joue un rôle important dans l'interaction via le HotBox.

Le HotBox est un menu 2D semi-transparent contenant plusieurs boutons ou items de menus. Pour afficher le HotBox, il faut appuyer la touche de MND et la garder enfoncée. Le HotBox est alors affiché par-dessus la fenêtre principale de l'application, et reste affichée tant et aussi longtemps que le bouton de la MND reste enfoncé. Pendant ce temps, la MD peut déplacer la souris, sélectionner des items dans le HotBox avec le curseur de la MD, et activer ces items, pour lancer des commandes ou ouvrir des sous-menus. En fait, la MD peut activer une suite de plusieurs commandes dans le HotBox, qui reste affiché jusqu'à ce que la MND relâche son bouton.

Il est important de noter que, lorsque le HotBox est affiché, il est initialement centré sur le curseur de la MD. Cela assure que, une fois que l'utilisateur est habitué au layout du HotBox, il pourra appuyer le bouton de MND et ensuite rapidement déplacer sa MD avec un mouvement relatif au centre, pour arriver par-dessus un item désiré.

Le HotBox a été adapté par McGuffin et Jurisica (2009) pour interagir avec des visualisations de graphes (Figure 1.9). Le HotBox de McGuffin et Jurisica (2009) a aussi été étendu pour permettre de cliquer (avec la MD) sur une commande, et ensuite de glisser pour fournir un

argument de 1D ou 2D pour la commande [2]. Par exemple, en cliquant sur la commande “Move” au centre du HotBox, l’utilisateur peut ensuite glisser pour fournir le déplacement désiré pour la commande. Par contre, si l’utilisateur clique sur la commande “Scale and Rotate”, un glissement aura pour effet de faire une rotation et un changement d’échelle (le déplacement en $x$ et en $y$ du glissement contrôlant chacun).

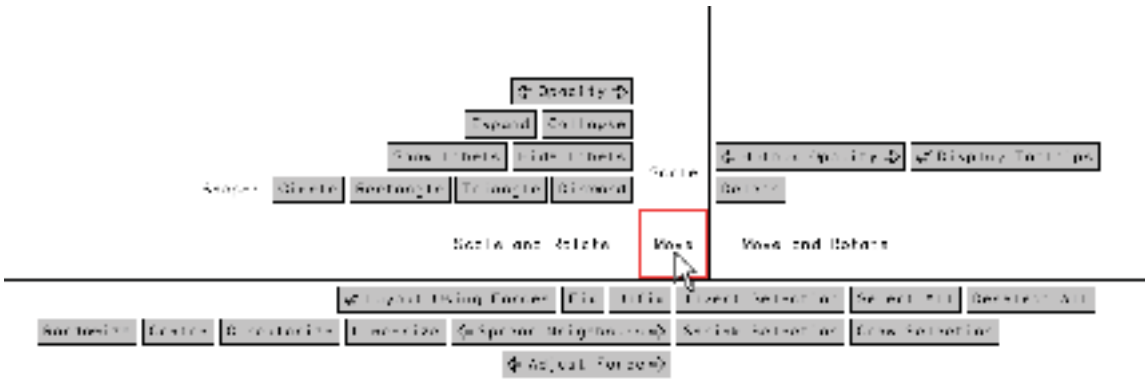

Figure 1.9 Un HotBox pour manipuler un graphe, tiré de McGuffin et Jurisica (2009)

Dû au fait que le HotBox peut occuper un grand espace, il peut contenir un grand nombre d’items, chacun pouvant être un point de lancement pour un sous-menu qui contient par la suite plusieurs items lui-même. Donc, le HotBox permet d’accéder à des centaines, voir même des milliers d’items, avec une interaction simple. De plus, le HotBox a les avantages des widgets contextuels mentionnés dans la section 1.2.1 : il occupe de l’espace seulement lorsque la touche de la MND est appuyée, et élimine les mouvements aller-retour car il est affiché déjà centré sur le curseur.

Remarquez aussi que, contrairement au ToolGlass, le HotBox permet de seulement sélectionner des commandes et non des objets. Donc, typiquement, l’utilisateur aura à sélectionner un ou plusieurs objets (par exemple, avec une sélection en lasso), ensuite l’utilisateur pourra faire afficher le HotBox pour lancer des commandes.

2. Voir `http://www.youtube.com/watch?v=U2vhjdnSfsQ` pour une démonstration.

## 1.4 La visualisation des graphes

Un graphe est une structure générale qui est utilisée assez souvent pour la modélisation des relations entre les données avec une vaste application en bio-informatique, exploration du web, sociologie, etc. Un graphe est composé de deux types d'éléments natifs : les nœuds et les arêtes. Formellement, un graphe est un couple [3] $G = (V, E)$ où $V$ est un ensemble fini de nœuds et $E$ est l'ensemble des arêtes. Si le graphe n'est pas orienté, $E$ est l'ensemble de paires [4] $\{x, y\}$, où $x, y \in V$ ; et si le graphe est orienté, $E$ est l'ensemble de couples $(x, y)$, où $x, y \in V$.

Chaque nœud peut posséder plusieurs attributs, par exemple une étiquette, laquelle selon le domaine d'application pourrait prendre sa propre signification. Par exemple, si le graphe représente un composé chimique, alors l'étiquette du nœud peut représenter un type d'atome. Du côté des arêtes, ils peuvent aussi porter des attributs, par exemple leur direction, ou de la même façon qu'un nœud : une étiquette.

### 1.4.1 Interagir avec des visualisations de graphes

Il existe plusieurs algorithmes pour calculer le *layout* (disposition) d'un graphe (Di Battista *et al.*, 1999; Kaufmann et Wagner, 2001). Une fois un layout calculé, plusieurs tâches d'exploration peuvent être effectuées, telles que décrites dans la taxonomie de Lee *et al.* (2006). Des exemples de tâches dans cette taxonomie sont : trouver les voisins d'un nœud donné, trouver le nœud avec le plus grand degré (c.-à-d. le plus grand nombre de voisins), trouver un nœud avec un attribut donné, et suivre un chemin (suite de nœuds adjacents) quelconque. En gros, il s'agit de tâches de requête et de navigation topologique. Quelques widgets spécialisés ont été proposés pour aider avec ces tâches (Moscovich *et al.*, 2009; Bezerianos *et al.*, 2010b; Viau *et al.*, 2010), mais aucun bimanuel.

Il y a aussi des tâches, non identifiées dans la taxonomie de Lee *et al.* (2006), qui sont nécessaires pour modifier le layout et la présentation graphique d'un graphe. Ces tâches peuvent être

3. En anglais, un *ordered pair*.
4. En anglais, un *unordered pair*.

réduites en termes de sélections d'objets (de nœuds et/ou d'arêtes) et de sélections de commandes, tel que discuté dans la section 1.2.1. Un exemple d'une suite de sélections d'objets et de commandes est présenté dans la Figure 1.10.

Encore d'autres tâches possibles sont reliées au choix de présentation graphique d'un graphe. Pour bien comprendre ces choix, les sections suivantes présentent différentes façons de visualiser un graphe.

### 1.4.2 Représentation nœuds-liens

La représentation la plus habituelle des graphes est le diagramme nœuds-liens (Figure 1.11). Elle se sert de formes géométriques pour représenter les nœuds (cercles, triangles, carrés etc.) et des lignes ou de courbes pour les arêtes. Les croisements d'arêtes deviennent un problème pour l'interprétation dès que le nombre de nœuds et arêtes visualisés augmente (Holten et Van Wijk, 2009). La représentation nœuds-liens est conseillée pour les graphes ayant moins d'une vingtaine de sommets (Ghoniem *et al.*, 2004b). Elle est intéressante quand les chemins entre deux ou plusieurs nœuds doivent être compris (Shen et Ma, 2007).

### 1.4.3 Représentation matricielle

Une façon alternative de visualiser un graphe est sous forme de sa matrice d'adjacence (Ghoniem *et al.*, 2004a) (Figure 1.12, droite). Cela peut sembler bizarre à première vue, mais la représentation matricielle offre l'avantage d'éliminer entièrement les croisements d'arêtes, ce qui peut être un avantage important quand le graphe a beaucoup de liens et qu'on veut voir clairement quelles arêtes sont présentes et lesquelles sont absentes.

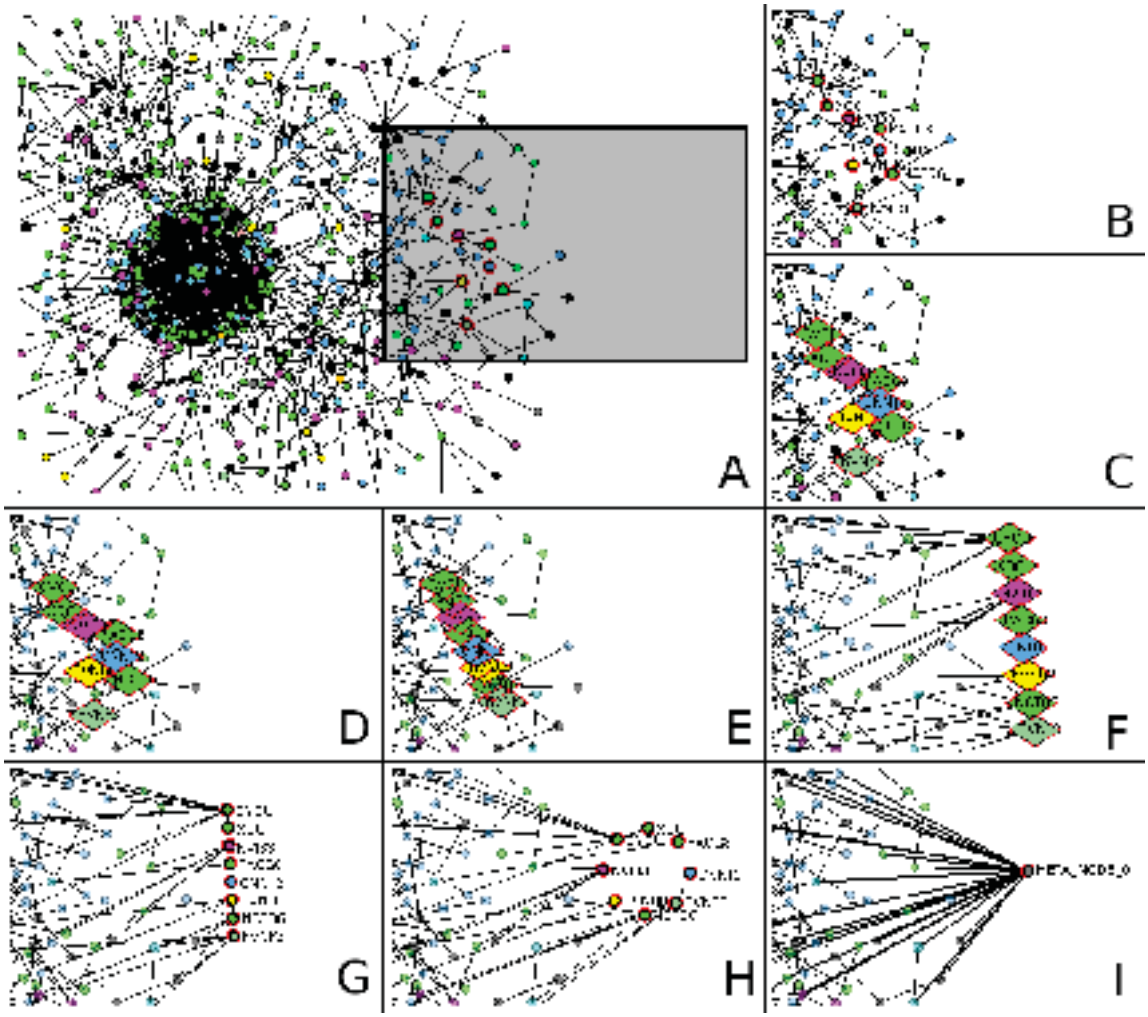


Figure 1.10 Une suite de manipulations sur un graphe. A : Un layout initial. L'utilisateur sélectionne 8 nœuds dans la région encadrée en gris, qui est montrée dans les cadres suivants. B : L'utilisateur lance une commande pour afficher les étiquettes des nœuds sélectionnés. C : Une commande est lancée pour changer la forme des nœuds. D : La sélection des nœuds est inversée, ensuite la transparence des autres nœuds est augmentée. E : Les positions des 8 sont projetées sur une ligne droite. F : L'ensemble des 8 nœuds est déplacé. G : Une commande est lancée pour revenir aux formes en cercles. H : Un layout en cercle est appliqué. I : Les 8 sont rassemblés en un seul méta-nœud.

La matrice d'adjacence est une matrice de booléens, où les colonnes et les rangées symbolisent les nœuds du graphe. Quand elle sert à montrer un graphe orienté, les colonnes représentent l'origine des arêtes et les rangées représentent le point final vers les nœuds. Pour les graphes non orientés, la matrice est symétrique. La cellule d'intersection entre la rangée et la colonne contient la valeur "vraie" (1) s'il existe une connexion entre tous les deux nœuds, au

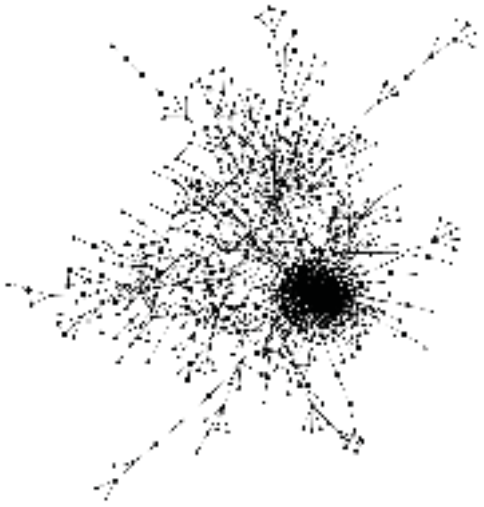

Figure 1.11 Un graphe sous forme nœuds-liens, dont le layout a été déterminé par un algorithme de disposition par simulation de forces de ressort et de répulsion.


Figure 1.12 À gauche, la représentation nœuds-liens d'un graphe. À droite, la représentation en matrice d'adjacence du même graphe. Remarquez, par exemple, que le nœud F est adjacent aux nœuds D, E, et H dans les deux représentations.

cas contraire la valeur contenue est fausse (0). La valeur booléenne peut être remplacée par une valeur d'attributs associés avec les nœuds précisant un peu plus d'information visuelle (Ghoniem *et al.*, 2004b). Chaque cellule correspondant à la diagonale de la matrice représente un lien avec le même nœud. Généralement ce type de liens sont ignorés, alors les valeurs dans les cellules de la diagonale ne sont que des zéros.

Comme chaque arête est définie par elle-même, dans un espace non partagé, il n'y en a pas de problèmes d'occlusion. Alors, les matrices présentent les caractéristiques suivantes :

a. Élimination des croisements d'arêtes

b. Représentation visuelle uniforme (Van Ham, 2003).

c. Les matrices visualisent facilement l'absence des liens entre deux éléments (Otjacques et Feltz, 2005).

Malgré ces avantages, un gros désavantage de la représentation matricielle est qu'il est très difficile de voir des chemins sur plusieurs nœuds dans une matrice. Dans une représentation nœuds-liens, par contre, les chemins sont relativement faciles à voir. Pour combiner les avantages des deux représentations, Henry *et al.* (2007) ont proposé NodeTrix, une visualisation hybride (Figure 1.13) permettant à l'utilisateur de visualiser différentes parties du graphe sous différentes formes. Typiquement, les parties denses du graphe sont visualisées sous forme matricielle pour éliminer les croisements d'arêtes et mettre en évidence les grappes (*clusters*), tandis que le reste du graphe est visualisé sous forme nœuds-liens. Cela offre plus de flexibilité à l'utilisateur, lui permettant des compromis différents dans différentes parties du graphe.

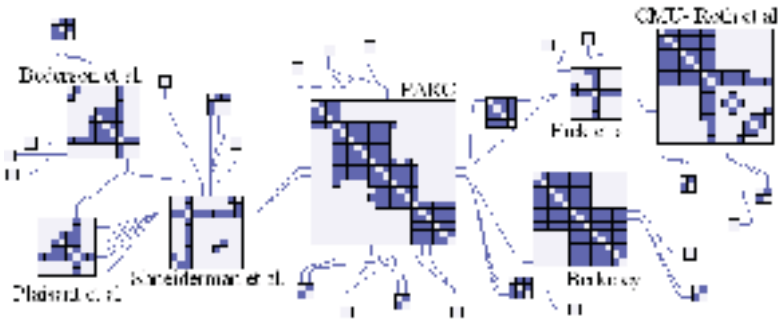

Figure 1.13 Une visualisation hybride d'un graphe, qui mélange les représentations en matrice et nœuds-liens. Cet hybride s'appelle NodeTrix (Henry *et al.*, 2007).

### 1.4.4 Représentation en nuages de points

Dans plusieurs domaines, les données sont souvent sous forme multidimensionnelle ou multivariée. Ces données peuvent être énumérées dans un tableau, avec une rangée pour chaque "point" ou $N$-uplet, et une colonne pour chaque dimension ou variable (Figure 1.14, gauche). Une approche générale pour visualiser ces données est d'utiliser une matrice de nuages de points, aussi appelée SPLOM (*ScatterPLOt Matrix*). Le SPLOM (Figure 1.14, droite) montre les nuages de points ayant toutes les paires d'axes possibles. Ce genre de matrice permet de rapidement voir s'il y a des corrélations entre des variables, de voir où il y a des données aberrantes (*outliers*), et de voir quelles paires de variables sont intéressantes.

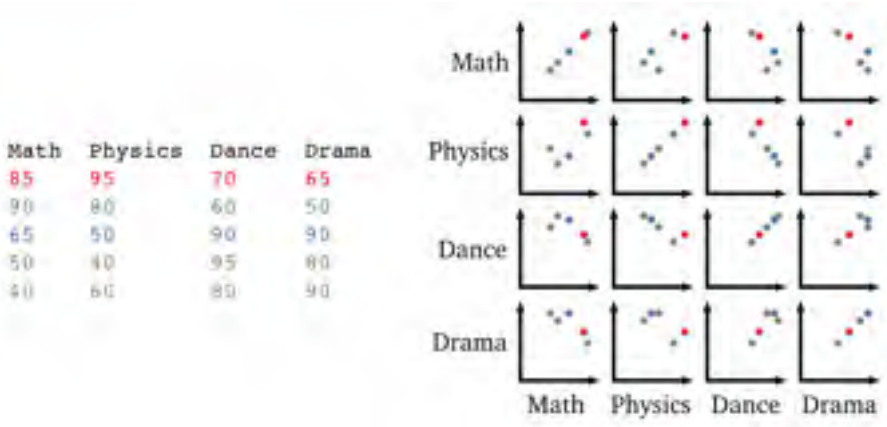


Figure 1.14 Un ensemble de données multidimensionnelles (où, plus précisément, de données multivariées). À gauche : les données sous forme de tableau. Chaque colonne correspond à une dimension (ou variable), chaque rangée correspond à un point (ou $n$-uplet). Dans cet exemple, chaque rangée est un étudiant, et chaque colonne est un cours pour lequel l'étudiant a une note. À droite : les données sous forme de matrice de nuages de points, ou SPLOM (*Scatter PLOt Matrix*).

Une approche semblable peut s'appliquer aux graphes, si on interprète les nœuds du graphe comme des $N$-uplets. Pour chaque nœud, on peut calculer des métriques, telles le degré, différentes sortes de centralité, le coefficient de regroupement (*clustering coefficient*), etc. Chaque

nœud peut aussi avoir différents attributs, comme un âge, sexe, revenus, etc. On peut alors interpréter les nœuds comme des $N$-uplets dont les coordonnées sont ces métriques et ces attributs, permettant de visualiser le graphe avec des nuages de points, ou même un SPLOM complet (Figure 1.15, droite).

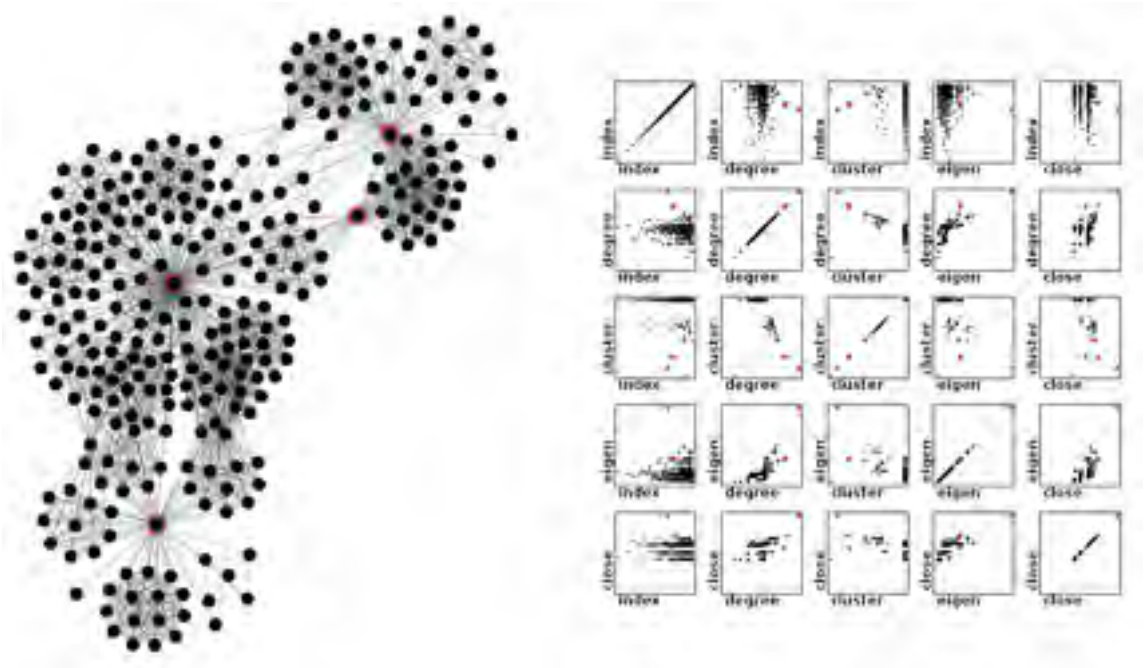


Figure 1.15 Deux représentations du même graphe. À gauche : diagramme nœuds-liens. À droite : matrice de nuages de points, avec les cinq dimensions indice, degré, coefficient de regroupement, centralité de *eigenvector*, centralité de *closeness*. Chaque nœud apparaît comme un point dans chaque nuage de points.

Des visualisations de graphe sous forme de SPLOM ont déjà été proposées (Bezerianos *et al.*, 2010a; Viau *et al.*, 2010). Par contre, une possibilité qui n'a pas encore été investiguée serait de mélanger les représentations nœuds-liens et nuages de points, pour faire une visualisation hybride analogue à NodeTrix (Figure 1.13)

## 1.5 Résumé

Nous avons vu que les widgets contextuels et l'entrée bimanuelle sont deux manières d'améliorer les GUIs. Notre but est d'améliorer les GUIs pour visualiser les graphes, alors on peut se demander à quel point les widgets contextuels et l'entrée bimanuelle ont été appliqués pour la visualisation interactive des graphes.

Certains widgets contextuels ont été conçus spécialement pour l'interaction avec les graphes (Moscovich *et al.*, 2009; Bezerianos *et al.*, 2010b; Viau *et al.*, 2010), mais ce sont des widgets uniquement unimanuels. Le ToolGlass de base a déjà été appliqué à l'édition de graphes (Beaudouin-Lafon *et al.*, 2001), sans innover au niveau du fonctionnement du ToolGlass. L'entrée multitactile a déjà été appliquée aux visualisations de graphes (Frisch *et al.*, 2009; Dwyer *et al.*, 2009; Schmidt *et al.*, 2010), par exemple en permettant la manipulation directe ou des gestes spécialisés d'édition, mais sans développer de nouvelle technique d'interaction *générique* qui permettrait d'exploiter plusieurs points d'entrée et de sélectionner une commande arbitraire parmi un grand ensemble de commandes.

De plus, une taxonomie de tâches (Lee *et al.*, 2006) pour interagir avec les graphes existe, mais comme nous avons vu, cette taxonomie n'est pas complète, car elle ne comprend pas les tâches d'édition de graphe ou de layout (par exemple, les tâches de la Figure 1.10), et ne comprend pas non plus de façons de passer entre les représentations nœuds-liens, matricielle, ou en nuages de points. Par contre, toutes les tâches à faire avec les graphes peuvent être réduites à des sélections d'objets (nœuds ou sous-graphes) et des sélections de commandes. Donc, les techniques d'interaction comme le HotBox ou le ToolGlass pourraient fournir une façon générique de lancer les commandes associées à ces tâches.

Parmi les widgets et les techniques d'interaction antérieures, ceux qui sont les plus génériques (permettant de lancer beaucoup de commandes différentes) et qui exploitent les deux mains sont le HotBox et le ToolGlass. Toutefois, il reste des questions ouvertes à répondre pour mieux appliquer ces techniques aux visualisations de graphes. Par exemple : est-il raisonnable de demander à l'utilisateur de faire un clic-à-travers un ToolGlass sur un seul nœud, même

si le nœud peut être très petit ? Ou y a-t-il une façon de rendre ces sélections plus faciles ? Comment les utilisateurs doivent-ils sélectionner des sous-ensembles de nœuds (sous-graphes) en utilisant des interactions bimanuelles ou multitactiles ? Entre le HotBox et le ToolGlass, lequel est meilleur ? Y a-t-il différents avantages et inconvénients du HotBox et du ToolGlass, et dans ce cas, faut-il chercher une façon de combiner les deux en un hybride ?

## 1.6 Objectifs

L'objectif général des recherches dans ce mémoire est d'améliorer les GUIs pour visualiser les graphes, en utilisant l'entrée bimanuelle. Plus particulièrement, nous cherchons

- Une méthode générique pour sélectionner des sous-ensembles de nœuds. Cette méthode devrait permettre de sélectionner des sous-graphes pré-calculés, devrait être extensible, et devrait permettre de sélectionner des chemins entre des nœuds en exploitant les deux mains.
- Une méthode générique de lancer des commandes. Cette méthode devrait permettre de lancer plusieurs commandes différentes, et permettre de fournir des arguments aux commandes en utilisant les deux mains, pour aller chercher tous les degrés de liberté des mains (car certaines commandes pourraient bénéficier d'arguments 2D ou même 4D). Cette méthode devrait aussi être extensible à un grand nombre de commandes. Idéalement, cette méthode devrait combiner les avantages des méthodes antérieures, telles le HotBox et/ou le ToolGlass.
- Des nouvelles idées de techniques d'interaction bimanuelles avec les graphes, en travaillant par analogie avec des techniques existantes.

Le prochain chapitre présente une nouvelle taxonomie d'interactions bimanuelles possibles sur les visualisations de graphes, et se sert de cette taxonomie pour donner un contexte à quatre nouvelles techniques d'interaction bimanuelle, qui poussent les travaux antérieurs plus loin et qui répondent aux objectifs ci-dessus. Parmi ces quatre nouvelles techniques, nous verrons le HotGlass, une combinaison hybride du ToolGlass et du HotBox.

# CHAPITRE 2

# NOUVELLES TECHNIQUES D'INTERACTION BIMANUELLE

## 2.1 Taxonomie d'interactions bimanuelles pour les visualisations

Pour comprendre les techniques d'interaction bimanuelle qui sont possibles avec les visualisations de graphes, une façon d'analyser les techniques possibles consiste à énumérer les rôles qui peuvent être joués par chaque main. Par exemple, dans le cas de ToolGlass, avant de faire un clic-à-travers, la MND positionne la palette d'outils en dessous du curseur de la MD. Donc, la MND sélectionne la *commande*. La MD, par contre, positionne le curseur par dessus l'objet, et sélectionne donc l'*objet*. Dans le cas de manipulation directe (Figure 1.6), on peut considérer que les deux mains servent à sélectionner la *commande*, car les deux spécifient la translation, rotation, et changement d'échelle à effectuer. Dans la section 1.3.2.1, nous avons aussi vu que les deux mains peuvent être déplacées pour effectuer un zoom ou une translation de *caméra*. Finalement, il y a des techniques d'interaction autour des métaphores de *lentilles* (comme les lentilles magiques (Bier *et al.*, 1993)), où une main déplace une lentille qui change la visualisation des objets en dessous.

Voilà donc 4 rôles qui peuvent être joués par chaque main : manipulation de *caméra*, sélection d'*objets* (nœuds ou sous-graphes), sélection de *commandes*, et manipulation de *lentille*. La Figure 2.1 présente une taxonomie de techniques d'interaction bimanuelle, classées selon le rôle joué par chaque main. Les techniques déjà publiées sont accompagnées d'une référence. Certaines sont accompagnées aussi d'un exposant, avec les significations suivantes :

1 La technique est déjà connue, et a aussi été réalisée dans le prototype de ce chapitre. Il y a 3 techniques de cette sorte.

2 La technique est novatrice, et a aussi été réalisée dans le prototype de ce chapitre. Il y a 4 techniques de cette sorte.

3 La technique est novatrice, mais n'a pas été réalisée. Il y a 1 technique comme ça.

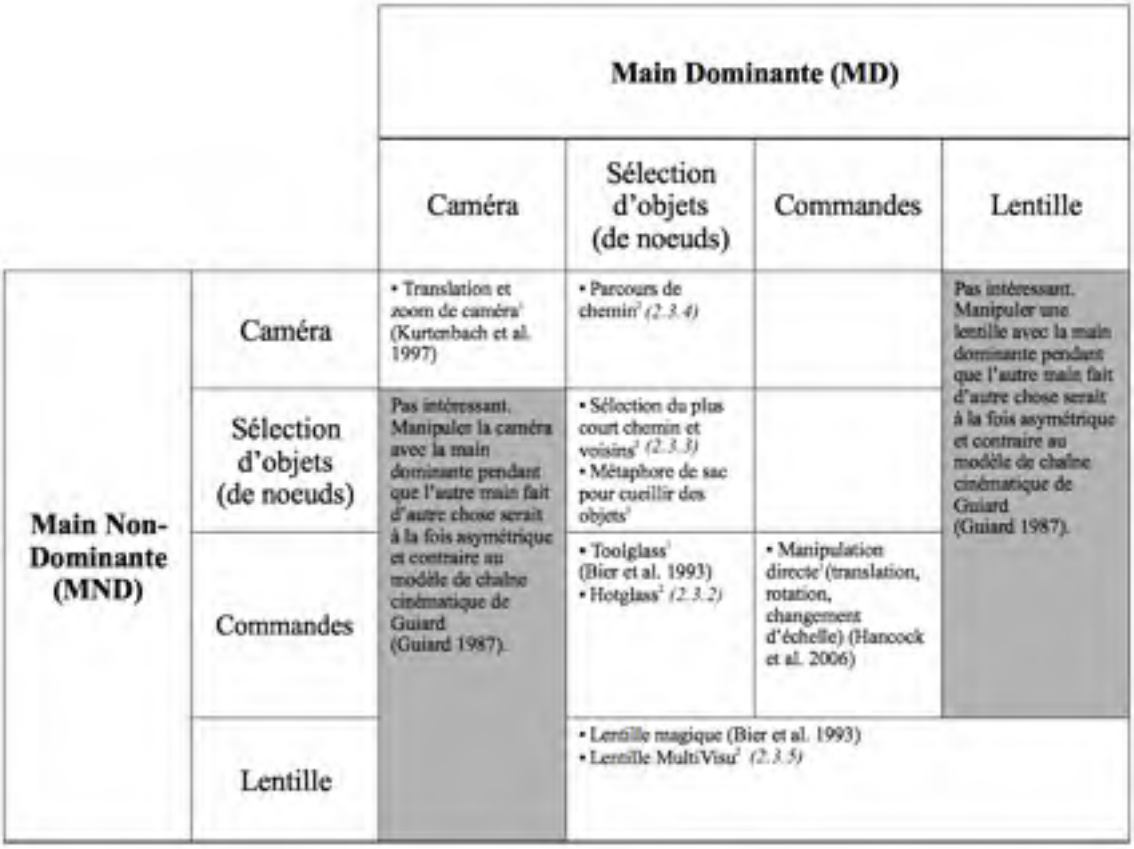


Figure 2.1 Taxonomie de techniques d'interaction bimanuelles. Remarquez que les techniques d'interaction symétriques se trouvent nécessairement sur la diagonale de cette taxonomie, tandis que les techniques asymétriques peuvent être sur la diagonale où bien ailleurs. Les exposants sur certaines techniques indiquent : (1) une technique déjà publiée, (2) une technique novatrice réalisée dans notre prototype, (3) une technique novatrice mais que nous n'avons pas réalisée.

Le seul exemple du troisième cas est la "Métaphore de sac pour cueillir des objets". Il s'agit d'une idée pour aider à sélectionner des ensembles de nœuds : la MND pourrait tenir un "sac" virtuel, et la MD pourrait saisir des nœuds et les mettre dans le sac pour définir un ensemble de nœuds sélectionnés. Cette idée de technique n'a pas été réalisée, mais pourrait être utile si l'utilisateur voulait gérer plusieurs groupes de nœuds ("sacs") sélectionnables.

Si la technique d'interaction bimanuelle est symétrique, évidemment, les deux mains jouent le même rôle, et la technique se trouve sur la diagonale de la taxonomie. Si la technique est asymétrique, elle peut se trouver soit hors de la diagonale (si les deux mains jouent des rôles différents) ou bien sur la diagonale (si les deux mains jouent le même rôle, mais de manière différente, comme dans le cas de la métaphore de sac : les deux mains servent à sélectionner des objets, mais de manières différentes).

Remarquons aussi que, concernant les cases de la taxonomie hors diagonale, qui sont nécessairement asymétriques, il y a certaines possibilités qui contredisent le modèle de chaîne cinématique de Guiard (section 1.3.3.1), et donc peuvent être éliminées. Ces parties de la taxonomie sont grisées. Par exemple, il ne serait pas logique d'avoir une technique où la MD manipule la caméra pendant que l'autre main sélectionne des objets. Le contraire, toutefois, pourrait être logique.

Finalement, remarquez que les 4 techniques novatrices réalisées pour ce mémoire (notées avec l'exposant 2) couvrent les 4 rôles possibles des mains :

- Notre technique de "parcours de chemin" (section 2.3.4) se sert de la MND pour manipuler la *caméra* pendant que l'autre main sélectionne des *objets*
- Notre technique de "sélection du plus court chemin" (section 2.3.3) est pour sélectionner des *objets*
- Notre technique de "HotGlass" (section 2.3.2) est pour lancer des *commandes*, et peut aussi servir à sélectionner des *objets*
- Notre technique de "lentille MultiVisu" (section 2.3.5) est une technique de *lentille*

## 2.2 Dispositifs matériels pour le prototype

Certaines des nouvelles techniques d'interaction proposées dans ce chapitre pourraient s'appliquer aux interfaces multitactiles. Mais, pour le prototype réalisé, nous avons décidé d'utiliser des dispositifs d'entrée bimanuelle indirecte, pour éviter les problèmes associés à l'entrée di-

recte, soient : l'occlusion de l'écran par les doigts, la manque de précision spatiale des doigts, la difficulté de distinguer l'identité des doigts, la difficulté de distinguer les états de survol et de glissement chez les doigts, et la difficulté de connaître l'orientation des bouts des doigts (voir la section 1.3.1).

Notre prototype est capable de lire des entrées de deux paires différentes de dispositifs, ce qui nous a permis de tester deux configurations matérielles. Premièrement, on peut utiliser deux souris conventionnelles à trois boutons chaque. Deuxièmement, on peut aussi utiliser deux dispositifs spécialement construits par l'auteur, ayant trois boutons chaque, et capables de capter la position et l'orientation de chaque dispositif (Figure 2.2). Chacun de ces dispositifs est muni d'un capteur Patriot de Polhemus [1], localisant la position et l'orientation du dispositif en 3D par rapport à un émetteur. Les dispositifs reposent sur une surface horizontale et leur position $z$ n'est pas utilisée par le prototype logiciel, mais l'orientation autour de l'axe des $z$ peut servir, par exemple, à effectuer une rotation d'un objet lors d'une manipulation directe unimanuelle.

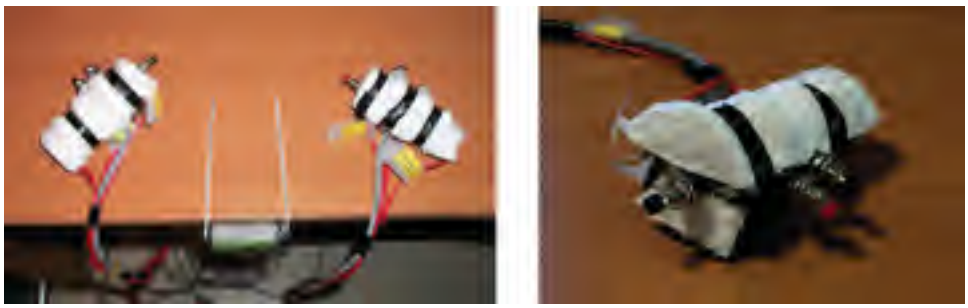

Figure 2.2 Des dispositifs de pointage spécialement construits pour le prototype. À gauche, les deux manettes avec l'émetteur (boîte grise) du Polhemus Patriot, qui sert comme origine pour le système de coordonnées. À droite, la manette gauche, montrant ses trois boutons.

Pour construire les dispositifs, des moules en argile et pâte à modeler ont été créées, puis elles ont été recouvertes d'une couche de vaseline, puis injectées de silicone. La couche de vaseline servait à éviter que le silicone colle avec les moules lors de la solidification, et a

1. http://polhemus.com/?page=Motion_Patriot

permis d'enlever le silicone solide. Les capteurs Patriot ont été incrustés dans les morceaux de silicone solide, pour lire les 6 degrés de liberté du dispositif (position et orientation). Les boutons de ces dispositifs sont connectés par des longs fils à deux souris USB normales, dont les boutons ont été enlevés. Donc, les événements de boutons des dispositifs sont toujours lus comme des événements de boutons de souris, que ce soit des souris normales ou les dispositifs Patriot qui sont utilisés.

## 2.3 Le prototype logiciel

Le prototype est écrit en Java, et les événements de deux souris (boutons et, si nécessaire, positions) sont obtenus avec la librairie JInput [2], permettant de distinguer deux souris USB connectées en même temps.

Si les dispositifs Patriot (Figure 2.2) sont utilisés, il faut aussi lire les positions et les orientations des capteurs Patriot. Pour faire cela, un utilitaire C++ est utilisé, qui boucle en lisant les positions et orientations actuelles et les écrit sur le *stardard output*, qui est ensuite lu comme *standard input* par l'application Java dans un fil d'exécution en arrière-plan.

Le prototype est fonctionnel sur les systèmes d'exploitation Linux et MS Windows, avec soit deux souris ou les deux dispositifs Patriot.

Comme déjà mentionné, les dispositifs Patriot captent leur orientation, contrairement aux souris qui ne captent que leur position. Une autre différence entre les deux sortes de dispositifs est que les souris standards détectent des déplacements relatifs, tandis que les dispositifs Patriot captent des positions absolues. Bien qu'il serait possible de détecter quand l'utilisateur lève les dispositifs de la surface (en lisant leur position $z$) et de simuler un "embrayage" de souris et calculer des déplacements relatifs, nous avons décidé de plutôt utiliser les positions absolues des dispositifs Patriot pour positionner leurs curseurs, pour mieux comprendre les avantages et les inconvénients du positionnement relatif (souris) versus absolu (Patriot). Toutefois, avec les dispositifs Patriot, il fallait rajouter un décalage (*offset*) fixe dans le sens horizontal entre les

2. `http://jinput.dev.java.net/`

systèmes de coordonnées des deux mains, pour éviter que les mains entrent en collision physique lorsque leurs curseurs deviennent très près l'un de l'autre. Avec ce décalage, quand les deux curseurs coïncident en position, les mains sont encore séparées de quelques centimètres.

### 2.3.1 Fonctionnalités de base

Cette section présente plusieurs fonctionnalités de base du logiciel : la manipulation directe, manipulation de caméra, sélection de nœuds, définition de sous-ensembles, simulation de forces de ressort.

Que les souris ou les dispositifs Patriot soient utilisés, chaque main possédera 3 boutons, que nous numérotons 1, 2, 3 de l'intérieur vers l'extérieur. Les trois boutons de la MD s'appellent MD.1, MD.2 et MD.3 (bouton interne, milieu, et externe, respectivement), et pareillement pour la MND : MND.1, MND.2, MND.3.

Étant donné que la manipulation de la caméra et la manipulation directe du réseau (c.-à-d. saisir et déplacer des nœuds) sont des opérations fréquemment réalisées, il y a des boutons réservés juste pour ces opérations : MD.3, MD.2 et MND.2.

Quand MD.3 est appuyé, l'utilisateur peut faire une translation et zoom bimanuel de caméra.

En utilisant MD.2 et MND.2 (boutons du milieu), l'utilisateur peut saisir des nœuds individuellement ou bien des ensembles de nœuds (Figure 2.3) et les déplacer. Chaque main peut saisir un nœud ou un ensemble, et le déplacer indépendamment de l'autre main. Si l'utilisateur clique là où il y a un chevauchement entre plusieurs ensembles, l'ensemble avec le périmètre le plus petit sera saisit, comme s'il était par dessus les autres. Si les dispositifs Patriot sont utilisés, une seule main peut aussi tourner l'ensemble saisit (tout comme dans le système de Kurtenbach *et al.* (1997)). Une innovation mineure que nous avons réalisée est que, quand l'utilisateur tourne les dispositifs, on multiplie l'angle du dispositif d'un facteur de quatre avant d'appliquer la rotation à l'ensemble. Ceci permet à l'utilisateur de tourner complètement (360°) l'ensemble de nœuds en appliquant juste une rotation confortable de la main de 90°. Les deux mains

peuvent également saisir un ensemble commun (Figure 2.3, en bas) et effectuer simultanément la translation, la rotation, et le changement d'échelle de cet ensemble.

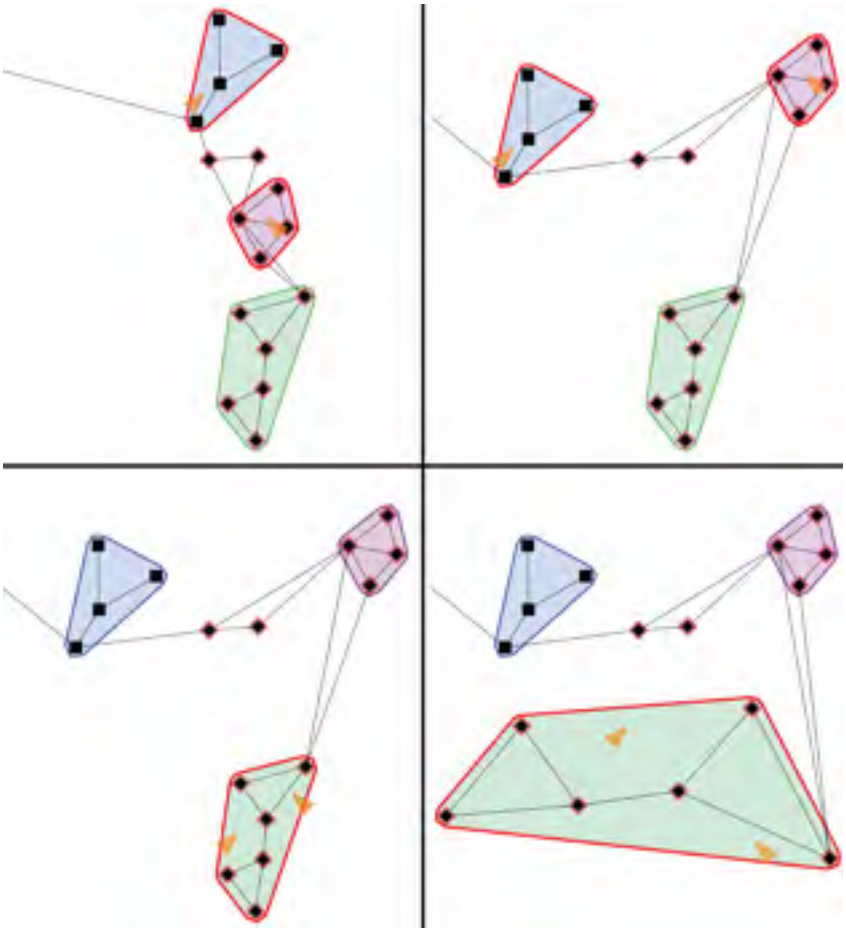

Figure 2.3 Manipulation directe des sous-graphes avec les deux mains. Ci-dessus, la translation de deux sous-graphes, chacun saisi dans une main. Ci-dessous, la translation, rotation, et changement d'échelle d'un seul sous-graphe par les deux mains.

Lorsque l'utilisateur manipule la caméra (MD.3), ou bien qu'il utilise les deux mains à la fois pour manipuler un seul ensemble (MD.2 + MND.2 sur un seul ensemble), le prototype permet une interaction bimanuelle qui est soit symétrique ou asymétrique, selon les options choisies. L'interaction bimanuelle symétrique dans ces cas correspond au geste bien connu de "pincement" dans les interfaces multitactiles, où les deux pointeurs semblent traîner des points

fixes de la surface, provoquant une contraction s'ils sont rapprochés. L'interaction symétrique semblait être l'approche évidente à utiliser au départ, mais nous avons vite remarqué des inconvénients avec le contrôle bimanuel symétrique. Lorsque l'utilisateur souhaite simplement déplacer la camera sans zoom, ou souhaite déplacer un ensemble sans le tourner ou changer son échelle, ce geste est très difficile à faire avec une interaction bimanuelle symétrique, car il nécessite de déplacer les mains ensemble et parallèlement avec la même vitesse et la même direction. Toute différence dans la vitesse des mains provoque soit une rotation, un changement d'échelle, ou les deux. (Ceci est également vrai pour le geste de pincement avec deux doigts, mais il semble être beaucoup moins problématique dans une interface multitactile, car il est beaucoup plus facile de déplacer deux doigts de la même main ensemble sans changer la distance entre eux.) Alors, dans notre prototype, le contrôle bimanuel asymétrique a été également implémenté comme option. Avec cette approche, le MD effectue la translation, et la MND effectue une rotation (lors du déplacement horizontal) et le changement d'échelle ou zoom (lors du déplacement vertical). Nous avons constaté que cette approche prend du temps pour s'y habituer, mais il permet que le déplacement soit fait en déplaçant simplement la MD, sans rotation ou changement d'échelle involontaire.

Dans le logiciel, nous faisons une distinction entre les notions de la *sélection* actuelle, et les *ensembles* de nœuds (aussi appelés *sous-graphes*). Il n'y a qu'une seule sélection de nœuds à la fois, montrée par une surbrillance rouge autour de chaque nœud (voir par exemple les Figure 2.7 B, C, D). Cependant, il peut exister plusieurs ensembles de nœuds, qui sont définis en mémoire, et affichés à l'utilisateur en dessinant des enveloppes convexes colorées autour des nœuds qui sont membres de l'ensemble. Deux nœuds individuels ainsi que les ensembles peuvent être saisis à l'aide des boutons de milieu (MD.2, MND.2), mais ceci ne les sélectionne pas ; ils ne sont que temporairement acquis pour la manipulation. La sélection des nœuds est faite en utilisant le bouton MD.1, par exemple, en dessinant un lasso ou un rectangle (Figure 2.4 et Figure 2.7 A). Une fois que des nœuds ont été sélectionnés, ils peuvent (optionnellement) être convertis en un ensemble, ce qui peut être utile pour faciliter leur acquisition plus tard.

Il y a plusieurs façons de sélectionner des nœuds avec MD.1. L'utilisateur peut cliquer directement sur un nœud individuel avec MD.1 pour basculer son état de sélection [3]. L'utilisateur peut aussi cliquer sur l'espace blanc avec MD.1 et ensuite glisser pour dessiner un rectangle ou un lasso de sélection (la forme de la traînée détermine si elle est interprétée comme un rectangle ou un lasso ; voir Figure 2.4 et Saund *et al.* (2003); McGuffin et Jurisica (2009) pour plus de détails). L'utilisateur peut également sélectionner des nœuds en cliquant directement sur un nœud avec MD.1 et en gardant le bouton enfoncé : cela fait afficher un menu contextuel spécialisé (Figures 2.10-2.11) qui sera expliqué plus tard, permettant de sélectionner un voisinage, un chemin, ou un sous-graphe pré-calculé contenant le nœud de départ.

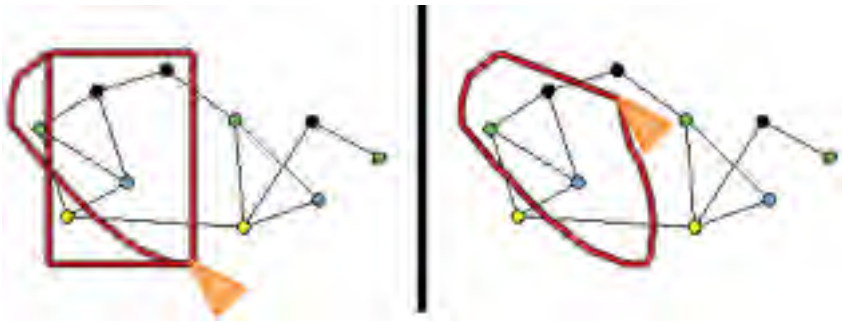

Figure 2.4 Le même bouton sert à sélectionner en rectangle ou en lasso, sans changement de mode. Gauche : L'utilisateur commence à glisser et le système dessine la tracée et le rectangle correspondant. Si l'utilisateur lâche le bouton, la sélection est interprétée comme un rectangle. Droite : L'utilisateur continue à glisser en revenant vers le point de départ. Une fois assez près du début du glissement, le système décide qu'il s'agit plutôt d'une sélection en lasso, et cesse de dessiner le rectangle.

L'utilisateur est libre de définir les ensembles de nœuds qu'il veut, en sélectionnant d'abord les nœuds, puis en convertissant la sélection en un ensemble. Chaque ensemble peut servir, en quelque sorte, comme un "signet" (*bookmark*) d'une sélection antérieure, qui peut être restaurée rapidement grâce à un menu contextuel sur l'ensemble, qui permet aux nœuds de l'ensemble d'être sélectionnés de nouveau. L'utilisateur peut également demander au système

3. Dans ce cas, l'action fonctionne comme un *toggle* : si le nœud n'est pas déjà sélectionné, alors il le sera après le clic, et vice versa.

d'afficher plusieurs ensembles pré-calculés (Figure 2.5) : il s'agit de toutes les arborescences, chaînes, "bicomposantes" (composantes biconnexes), couches de décomposition $K$-noyaux, et les ensembles de nœuds qui ont les mêmes voisins.

Figure 2.5 Sous-graphes pré-calculés par le prototype pour faciliter la saisie de différentes parties du graphe : chaînes (en bleu), arborescences (en mauve), bicomposantes (en orange), ensembles de nœuds ayant les mêmes voisins (en rose).

Tous ces ensembles peuvent être utilisés comme des poignées faciles à saisir pour déplacer et organiser la disposition des différentes parties du réseau. En même temps que l'utilisateur déplace des nœuds et des ensembles avec la manipulation directe, une simulation optionnelle de forces de ressort entre les nœuds peut mettre à jour de façon progressive les positions des autres nœuds. Cette simulation de forces se fait en arrière-plan par un autre fil d'exécution.

Voici un résumé du fonctionnement des boutons, dont certaines fonctionnalités seront expliquées dans les sections à venir.

| | |
|---|---|
| MD.1 | |
| Sur espace blanc : | trace une sélection en lasso ou en rectangle (Figure 2.4) |
| Sur le HotGlass : | lance une commande (Figures 2.6 A, B, C ; 2.7 C) |
| Sur un nœud : | ouvre le menu des (Figures 2.10-2.11) |
| Sur un ensemble : | ouvre un menu contextuel pour l'ensemble |
| MD.2 : | Manipulation directe (Figure 2.3) |
| MD.3 : | Translation et zoom bimanuel de la caméra |
| MND.1 : | Affiche une lentille MultiVisu (Figure 2.13) |
| MND.2 : | Manipulation directe (Figure 2.3) |
| MND.3 : | Téléporte le HotGlass (Figure 2.7 B) |

### 2.3.2 Le HotGlass

Tel que montré dans la section précédente, la manipulation directe permet d'effectuer des translations, rotations, et changements d'échelle de sous-ensembles de nœuds. Toutefois, il existe plusieurs autres commandes que l'utilisateur doit pouvoir exécuter : changer la forme des nœuds, montrer ou cacher les étiquettes des nœuds, retourner un ensemble de nœuds, changer les positions de nœuds pour les disposer le long d'une ligne ou sur un arc de cercle, supprimer ou ajouter des nœuds ou des liens, etc. La manipulation directe consomme déjà deux des six boutons matériels ; il n'y a pas assez de boutons pour réserver un pour chaque commande possible. Nous avons besoin d'une méthode générique de lancer des commandes, comme un menu. Deux candidats sont le HotBox et le ToolGlass, et nous aimerions savoir lequel serait mieux, ou s'il existe une autre technique encore mieux.

Le HotBox et le ToolGlass ont chacun plusieurs avantages : le HotBox est contextuel (et jouit donc de tous les avantages des widgets contextuels mentionés dans la section 1.2), le ToolGlass permet une sélection simultanée d'objet et de commande (ce qui risque d'être plus rapide). Les deux techniques sont aussi bimanuelles.

Il y a déjà eu des comparaisons expérimentales de la vitesse de différentes techniques d'interaction, pour comparer le ToolGlass avec d'autres techniques (Kabbash *et al.*, 1994; Leganchuk

*et al.*, 1998; Guimbretière *et al.*, 2005; Chen *et al.*, 2008). (Le HotBox, par contre, n'a jamais été évalué de façon expérimentale.) Ces études ont trouvé que le ToolGlass est parfois plus rapide que d'autres techniques, mais ont aussi trouvé que le ToolGlass peut être plus lent que des techniques simples et unimanuelles comme le Control Menu (Pook *et al.*, 2000; Chen *et al.*, 2008). Toutefois, le Control Menu a le désavantage, comme les Marking Menus, de pousser les concepteurs à organiser les commandes en groupements d'au plus 8 commandes (pour chaque sous-menu circulaire) et de se limiter à un maximum de quelques centaines de commandes (dû à un maximum de 3 niveaux de menus, avec 8 items par sous-menu (Kurtenbach et Buxton, 1993)). Ces limitations sont importantes lorsqu'on prend en considération que des logiciels de visualisation de graphes comme Tulip [4] et Cytoscape [5] comprennent un grand nombre de commandes complexes. Parmi toutes les techniques évaluées dans ces études, et discutées jusqu'à maintenant dans ce mémoire, il y en a trois qui sont uniques par le fait de permettre une organisation beaucoup plus flexible des commandes, et de permettre un accès à des *milliers* de commandes : ce sont la palette d'outils conventionnelle, le HotBox, et le ToolGlass. Ces trois techniques consistent toutes essentiellement en un menu 2D, dans lequel un concepteur peut organiser les items avec une disposition et des groupements arbitraires, et peut placer plusieurs sous-menus (comme des Marking Menus ou des Control Menus). On sait, des travaux de Kabbash *et al.* (1994), que la palette d'outils conventionnelle est plus lente que le ToolGlass. Le HotBox et le ToolGlass restent donc des candidats intéressants pour lancer des commandes dans notre prototype. Mais pouvons-nous faire mieux ?

Pour analyser les possibilités plus en détail, nous faisons une distinction entre trois genres de tâches. Les tâches Multi-Commandes (MC) sont des tâches où l'utilisateur applique une suite de plusieurs commandes différentes sur un seul objet. Une façon d'effectuer une tâche MC serait de sélectionner d'abord l'objet, et ensuite de sélectionner la suite de commandes dans un menu quelconque (comme un menu contextuel linéaire, ou bien dans un HotBox). Ensuite, les tâches Multi-Objets (MO) sont des tâches où l'utilisateur applique une commande sur plusieurs objets différents. Une façon d'effectuer une tâche MO serait de sélectionner le groupe d'objets,

4. `http://www.tulip-software.org/`
5. `http://www.cytoscape.org/`

et ensuite de lancer une seule commande qui est appliquée au groupe. Cette approche n'est toutefois pas possible si l'utilisateur veut appliquer la même commande *avec des arguments différents* à chaque objet. Dans ce cas, une approche typique serait de sélectionner un outil modal, comme un outil de translation, pour entrer dans le mode de cet outil, et ensuite d'appliquer l'outil à chaque objet désiré. Troisièmement, on a les tâches Multi-Commandes Multi-Objets (MCMO), qui impliquent une commande différente sur chacun d'une série d'objets.

Les comparaisons expérimentales déjà discutées (Kabbash *et al.*, 1994; Guimbretière *et al.*, 2005; Chen *et al.*, 2008) demandaient aux participants d'effectuer des tâches MCMO. Une autre comparaison, non expérimentale, est celle de Beaudouin-Lafon *et al.* (2001), qui dit que les Marking Menus sont bien pour les tâches MC, les palettes conventionnelles sont bien pour les tâches MO, et le ToolGlass est bien pour les tâches MCMO. Le travail de Beaudouin-Lafon *et al.* (2001) n'a toutefois pas discuté des HotBox. Nous remarquons aussi que, la raison pour laquelle Beaudouin-Lafon *et al.* (2001) semblent conseiller les palettes conventionnelles pour les tâches MO est que les tâches MO s'effectuent bien avec des outils modaux, qui se trouvent habituellement dans des palettes conventionnelles. Rien n'empêche, toutefois, de mettre des outils modaux à l'intérieur d'un ToolGlass ou à l'intérieur d'un HotBox. Pour ce qui est du conseil d'utiliser des Marking Menus pour les tâches MC, nous répétons que les Marking Menus sont limités en termes de flexibilité d'organisation des commandes et en nombre de commandes, et que le genre d'interaction nécessaire pour une tâche MC est très bien supporté par le HotBox : l'utilisateur sélectionne d'abord l'objet, ensuite fait afficher le HotBox pour lancer une série de commandes. Encore une fois, le HotBox et le ToolGlass ressortent comme des techniques avantageuses comparées aux autres.

Nous sommes d'accord avec Beaudouin-Lafon *et al.* (2001) que le ToolGlass est une technique particulièrement appropriée pour les tâches MCMO, car avec chaque clic-à-travers, l'utilisateur sélectionne à la fois un objet et une commande. De plus, nous faisons remarquer que le HotBox semble particulièrement approprié pour les tâches MC, car avec le HotBox, l'utilisateur a seulement besoin de sélectionner l'objet une fois. Les deux techniques, HotBox et

ToolGlass, peuvent aussi contenir des outils modaux et donc permettre d'effectuer des tâches MO facilement.

Y a-t-il une façon de combiner les avantages du ToolGlass et du HotBox ? Remarquons que, dans le cas de ToolGlass, la MND déplace la palette, tandis que dans le cas du HotBox, la MND ne fait qu'appuyer une touche de clavier pour faire apparaître le menu. Une idée pour les combiner, alors, serait d'avoir deux dispositifs de pointage, comme dans le cas d'un ToolGlass, et d'avoir un bouton réservé pour la MND qui joue le rôle de la touche de clavier dans le HotBox. Quand ce bouton de la MND n'est pas appuyé, la palette fonctionne comme un ToolGlass normal et permet des clics-à-travers par la MD (Figure 2.6). Par contre, quand le bouton de la MND est appuyé, la palette est "téléportée" en dessous de la MD (Figure 2.7 B), qui peut ensuite cliquer sur des commandes (Figure 2.7 C) pour les appliquer à la sélection antérieure, tout comme avec un HotBox. Voilà donc une façon de combiner les comportements de ToolGlass et de HotBox en une seule interface. L'utilisateur pourra alors choisir d'interagir avec des clics-à-travers ou suite à une téléportation, selon si la tâche à effectuer est une tâche MCMO ou une tâche MC. (Et pour les tâches MO, des outils modaux peuvent être placés dans la palette, accessible en tout temps, dans les états téléportés et non.)

Nous appelons cette fusion des deux techniques le HotGlass. La Figure 2.8 montre les commandes contenues dans le widget HotGlass réalisé dans le prototype. Des onglets séparent les commandes en trois groupes. Les commandes avec un astérisque à côté de leur nom (dans le premier onglet) peuvent être cliquées-à-travers (quand le HotGlass est utilisé comme un ToolGlass) et peuvent aussi être cliquées quand le HotGlass est téléporté, et dans ce deuxième cas la commande est appliquée à la sélection antérieure.

Pour téléporter le HotGlass, il faut appuyer MND.3 (bouton extérieur de la MND). Quand MND.3 est appuyé, le HotGlass saute pour être centré sous le curseur de la MD, et reste fixé en place (comme s'il était collé à l'écran), tant que la MND garde MND.3 enfoncé. L'utilisateur peut alors exécuter plusieurs commandes avec leur MD pendant une seule téléportation.

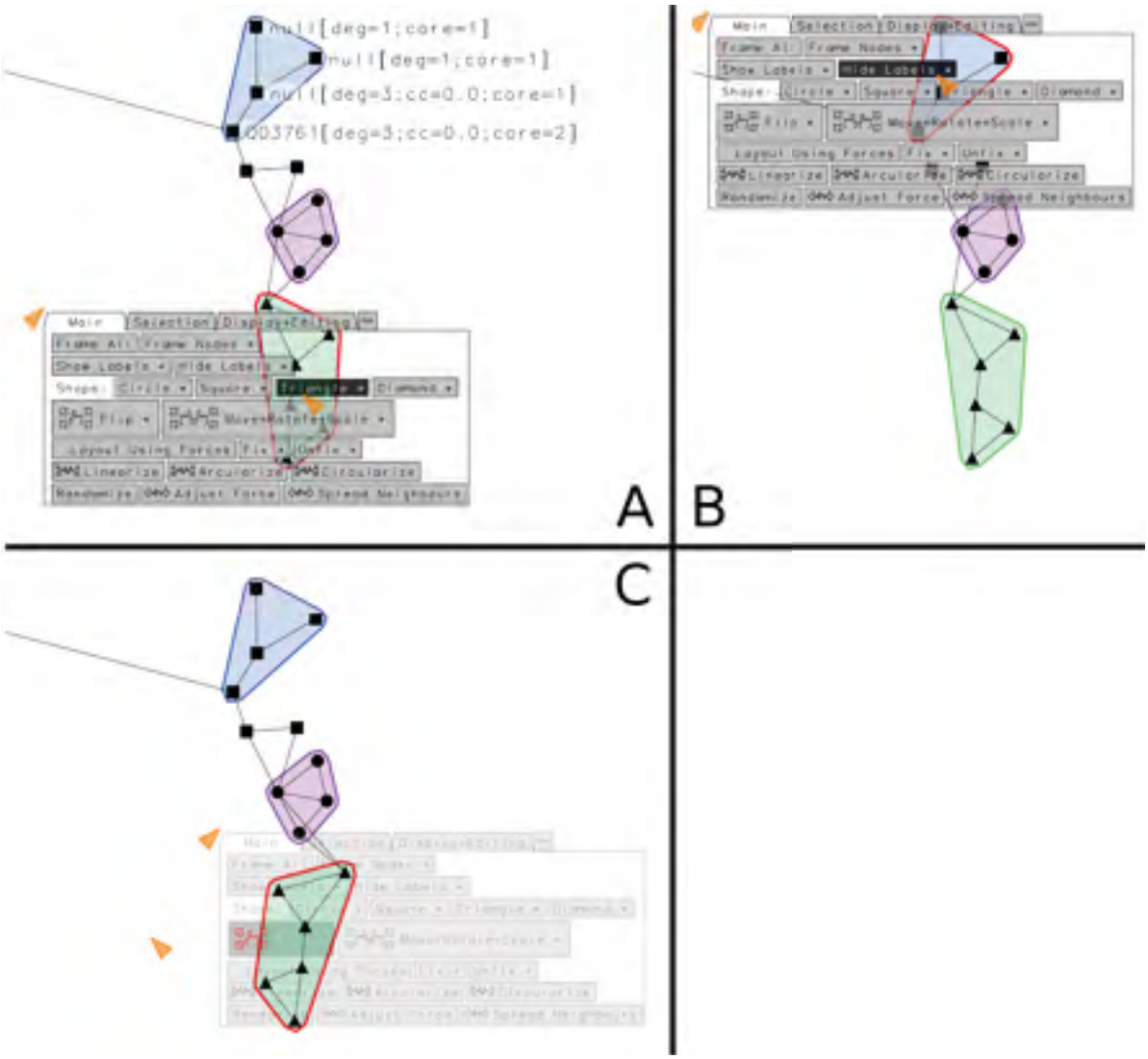


Figure 2.6 Cliquer-à-travers le HotGlass sélectionne à la fois une commande et le sous-graphe sur laquelle elle est appliquée, semblable au comportement d'un ToolGlass. A : L'utilisateur clique-à-travers la commande "Triangle" pour changer la forme des nœuds de l'ensemble vert. B : L'utilisateur clique-à-travers "Hide Labels" pour cacher les étiquettes de l'ensemble bleu. C : La MD clique-à-travers la commande "Flip" et glisse vers la gauche, pour retourner l'ensemble vert.

Pour cliquer sur une commande, que le HotBox soit téléporté ou non, on utilise MD.1. Tout comme le HotBox de McGuffin et Jurisica (2009), la MD peut garder le bouton MD.1 enfoncé et ensuite faire un glissement, pour fournir un argument avec la commande. Cependant, contrairement au HotBox de McGuffin et Jurisica (2009), les arguments peuvent avoir jusqu'à quatre degrés de liberté, puisque les deux mains sont disponibles pour glisser. Par exemple, dans le

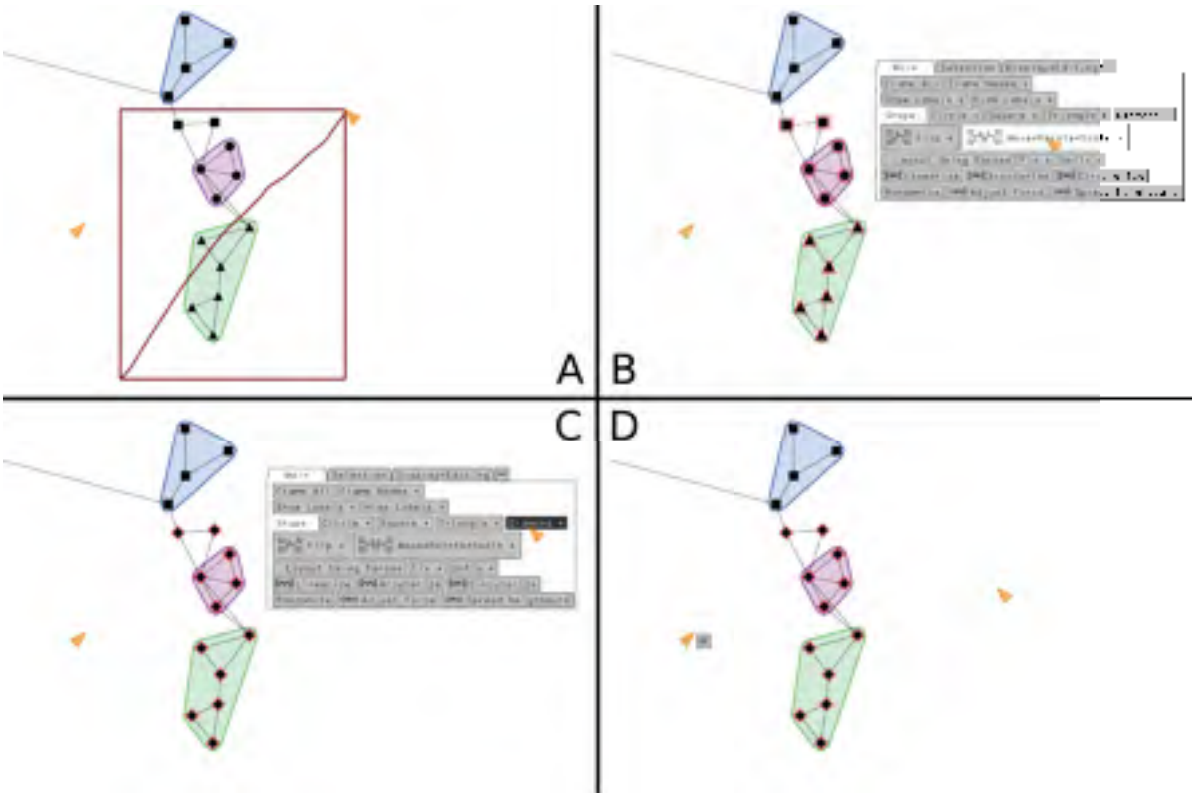


Figure 2.7 A : Sélection en rectangle de plusieurs nœuds (pendant le glissement de la MD, le HotGlass est automatiquement caché). B : les nœuds sélectionnés sont en surbrillance rouge, et le HotGlass est téléporté en dessous de la MD (initialement centré sur le curseur de la MD), permettant une suite de commandes d'être exécutées sur la sélection antérieure, semblable au comportement d'un HotBox. C : Toujours avec le HotGlass téléporté, l'utilisateur clique sur "Diamond" pour changer la forme des nœuds sélectionnés. D : L'utilisateur a enlevé la téléportation, et a cliqué sur le signe de moins à côté des onglets pour minimiser le HotGlass.

centre du HotBox (Figure 2.8) on retrouve la commande "Move + Rotate + Scale". Quand ce bouton est cliqué avec MD.1, les positions des deux curseurs sont utilisées pour spécifier les arguments à la commande.

Remarquez que certaines commandes dans le HotGlass sont d'envergure globale, telles que "Frame All", qui ajuste la caméra pour voir l'ensemble complet du réseau, ou la case à cocher "Layout Using Forces". Vu que ces commandes n'ont pas d'opérande particulier, elles fonctionnent de la même manière, que le HotGlass soit téléporté ou non.


Figure 2.8 Le HotGlass. A, B, et C montrent les trois onglets de boutons.

Le fait de pouvoir sélectionner un objet et ensuite téléporter le HotGlass pour lancer plusieurs commandes sur l'objet n'est pas juste utile pour les tâches MC. La téléportation est utile aussi quand l'ensemble de nœuds qu'on veut modifier est difficile à cliquer directement dessus avec un clic-à-travers. Par exemple, l'ensemble de nœuds pourrait être caché par d'autres ensembles,

ou pourrait être éparpillé au sein d'un enchevêtrement d'autres nœuds. Dans un tel cas, l'utilisateur peut sélectionner les nœuds par un autre moyen, comme le menu des Figures 2.10-2.11, puis téléporter le HotGlass pour pouvoir invoquer une ou plusieurs commandes sans défaire (ou risquer d'avoir à refaire) la sélection des nœuds.

Lorsque l'utilisateur souhaite translater, tourner, et/ou changer l'échelle d'un ensemble, il peut utiliser les boutons réservés pour la manipulation directe (MND.2 et MD.2) tels que discutés précédemment. Toutefois, ces opérations sont accessibles aussi dans le HotGlass, grâce au bouton "Move + Rotate + Scale". Ce bouton de HotGlass pourrait être très utile si la configuration matérielle ne permettait pas suffisamment de boutons pour en réserver deux pour la manipulation directe. Cependant, nous avons remarqué un problème avec le bouton "Move + Rotate + Scale" : quand le bouton est appuyé, soit en mode téléporté ou non, les positions initiales des curseurs peuvent être tout à fait inappropriées pour la manipulation bimanuelle symétrique. Lorsque l'utilisateur se sert des boutons MND.2 ou MD.2, l'utilisateur peut d'abord positionner confortablement (selon la forme de l'ensemble à manipuler) les curseurs avant de cliquer, mais ce n'est pas le cas avec le bouton "Move + Rotate + Scale" du HotGlass, parce que les positions initiales des curseurs sont déterminées par l'emplacement du bouton dans le HotGlass. Pour résoudre ce problème, nous avons implémenté un repositionnement (*warp*) automatique des curseurs (Figure 2.9). Lorsque l'utilisateur clique sur le bouton "Move + Rotate + Scale", le logiciel effectue une analyse en composantes principales (ACP, ou en anglais, *Principle Componant Analysis*) des positions des nœuds impliqués, pour trouver leur axe principal, ensuite, les curseurs sont repositionnés aux projections extrêmes des nœuds sur cet axe. Cela assure que les curseurs se trouvent aux extrémités de la sélection de nœuds, permettant une rotation et un changement d'échelle facile. Le repositionnement est choisi de telle sorte que le curseur gauche n'est jamais mis à droite du curseur droit. Une fois que l'utilisateur a terminé la manipulation des nœuds, et que le bouton MD.1 est relâché, une question qui reste est de savoir si le système devra remettre les positions des curseurs comme avant (c.-à-d. soustraire l'effet du repositionnement antérieur). Si des dispositifs de pointage relatifs sont utilisés, tels que les souris normales, nous ne conseillons pas de faire cela, puisque l'utilisateur sera

déjà adapté au premier repositionnement, et pourra être perturbé par un autre. Toutefois, pour les appareils avec des positions absolues comme les dispositifs Patriot, nous conseillons que le premier décalage introduit ne soit que temporaire, et soustrait au moment du relâchement, pour éviter un changement permanent dans le mappage spatial des dispositifs.

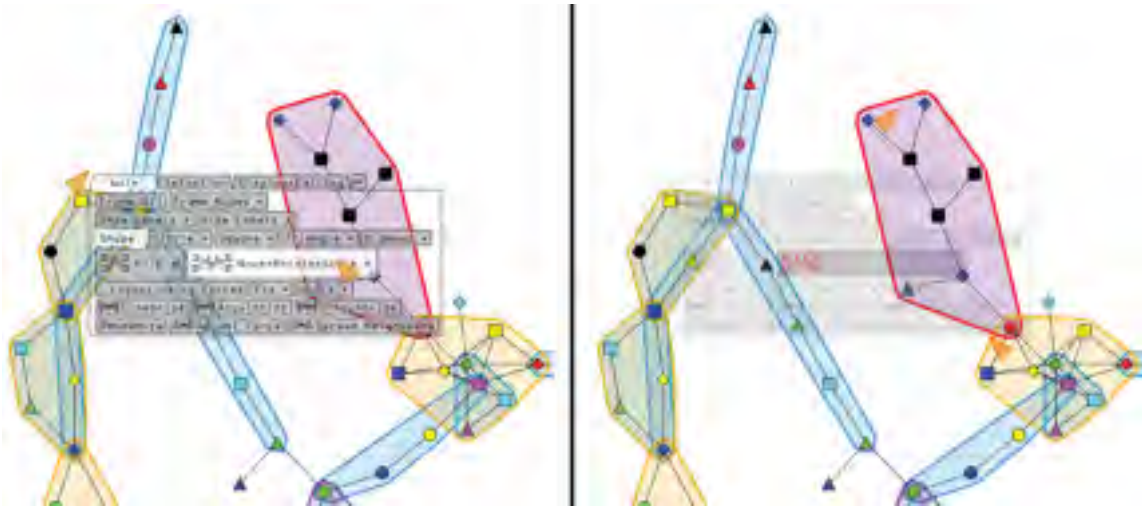

Figure 2.9 À gauche, remarquez les positions des curseurs avant de faire un clic-à-travers. À droite, après avoir fait un clic-à-travers du bouton “Move+Rotate+Scale” sur le sous-graphe mauve, les curseurs ont été repositionnés automatiquement aux extrémités du sous-graphe mauve, pour faciliter la manipulation directe.

Une amélioration possible au HotGlass qui n’a pas été réalisée est la possibilité de cliquer-à-travers non seulement sur des ensembles, mais aussi sur des nœuds individuels. (Actuellement, pour appliquer une commande sur un seul nœud, il faut soit utiliser la téléportation, ou bien créer un ensemble d’un seul nœud et cliquer-à-travers sur cet ensemble.) Bien que chaque nœud peut être petit, lorsque l’utilisateur clique sur un bouton et qu’il n’est pas sur un ensemble, le système pourrait supposer que l’utilisateur a l’intention d’effectuer la commande sur le nœud le plus proche. Un tel comportement, c’est-à-dire de prendre le nœud le plus proche, est semblable au bubble cursor (Grossman et Balakrishnan, 2005), et un tel curseur pourrait même être affiché

sous le HotGlass, à chaque fois que l'utilisateur a le curseur de la MD sur le HotGlass et non sur un ensemble.

D'autres questions de conception que nous n'avons pas complètement répondues sont reliées à la façon de rajouter plus de commandes au HotGlass. Tel que déjà mentionné, une façon de faire serait de rajouter des sous-menus contextuels à l'intérieur du HotGlass (en fait, notre HotGlass contient déjà des sous-menus linéaires permettant de sélectionner des sous-graphes pré-calculés). De plus, tel que montré dans la Figure 2.8, notre HotGlass est muni d'onglets. Nous avons observé que le fait de cliquer sur l'en-tête d'un onglet pour basculer vers cet onglet n'est pas une action très rapide, et brise la fluidité des autres interactions. Nous avons donc expérimenté avec une autre manière de basculer entre les onglets, en utilisant des événements de franchissement de bordures (*edge crossing*) (Apitz et Guimbretière, 2004). Dans une interface par franchissement, quand le curseur traverse une cible ou une frontière quelconque, cela peut déclencher une action. Dans le cas du HotGlass, nous avons rajouté une fonctionnalité optionnelle qui fait que, lorsqu'on glisse vers le haut sur l'en-tête d'un onglet, cela active l'onglet en question, sans avoir à cliquer sur l'en-tête. Ensuite, un mouvement de curseur vers le bas permet de revenir dans le HotGlass, sans changer d'onglet. Un tel mouvement aller-retour, vers l'extérieur ensuite vers l'intérieur, fait penser au FlowMenu de Guimbretière et Winograd (2000). Une autre possibilité serait de déclencher un changement d'onglet quand le curseur franchit une des bordures du bouton central ("Move + Rotate + Scale"). Quand le HotGlass est téléporté, initialement ce bouton est en dessous du curseur de la MD, alors un mouvement rapide de la MD dans une des quatre directions cardinales (nord, sud, est, ouest) pourrait servir à basculer vers un des onglets, un peu comme dans un menu circulaire à quatre items.

### 2.3.3 La sélection du plus court chemin et ses voisins

Le HotGlass présenté dans la dernière section est une manière générique de lancer des commandes en utilisant les deux mains. Nous avons également investigué des manières d'utiliser l'entrée bimanuelle pour sélectionner des ensembles de nœuds.

Comme chaque main contrôle un curseur et peut alors spécifier un nœud du graphe, il est assez naturel de penser à utiliser les deux mains pour sélectionner le plus court chemin entre deux nœuds. Pour rendre une telle technique de sélection plus interactive, on pourrait permettre à l'utilisateur de faire un survol de curseur avec une main (ou les deux), et de re-calculer le plus court chemin de façon dynamique, montrant le chemin en surbrillance, avant de compléter la sélection. Nous avons trouvé une façon d'intégrer cette fonctionnalité dans un menu contextuel qui permet aussi d'autres sortes de sélections de sous-graphes (Figures 2.10-2.11).

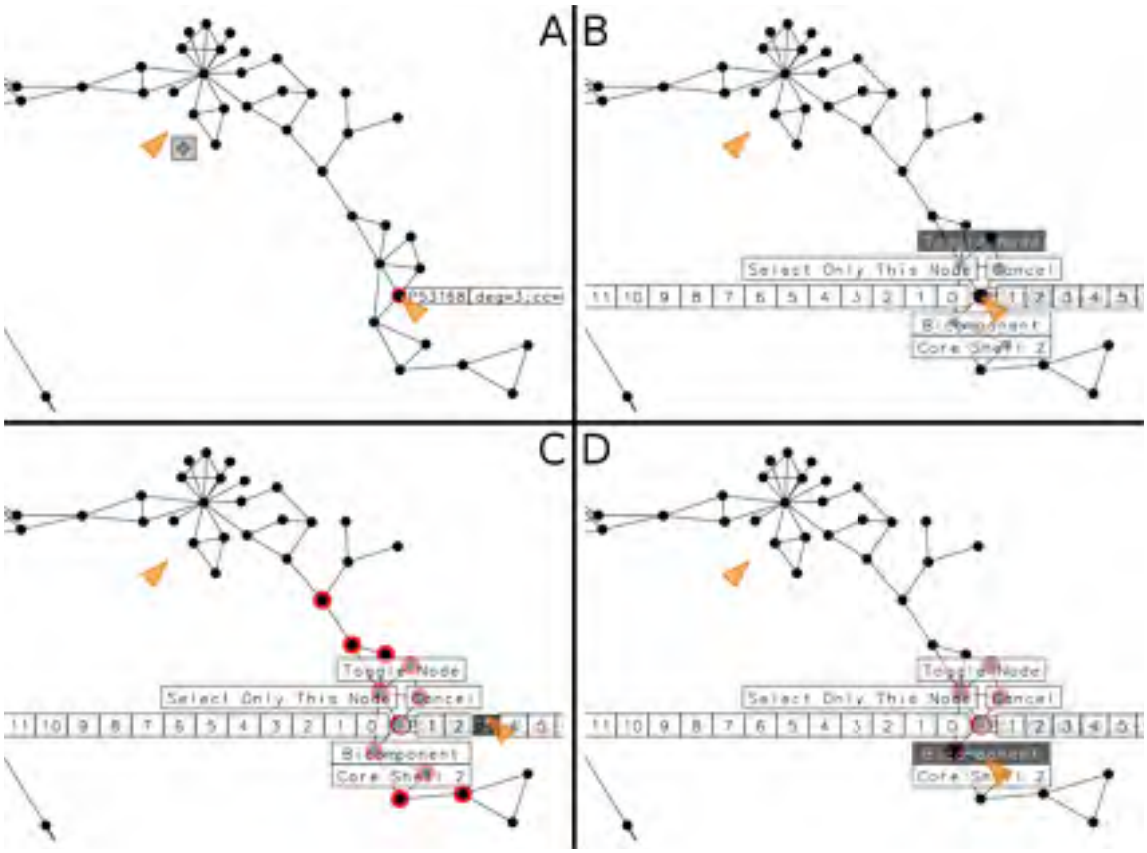


Figure 2.10 Un menu contextuel pour effectuer différentes sortes de sélections. A : La MD sur un nœud, avec infobulle. B : La MD clique sur le nœud et active le menu contextuel. C : Sélection des voisins du nœud jusqu'à une distance de 3 arêtes. D : Sélection d'une bicomposante dont le nœud fait partie.

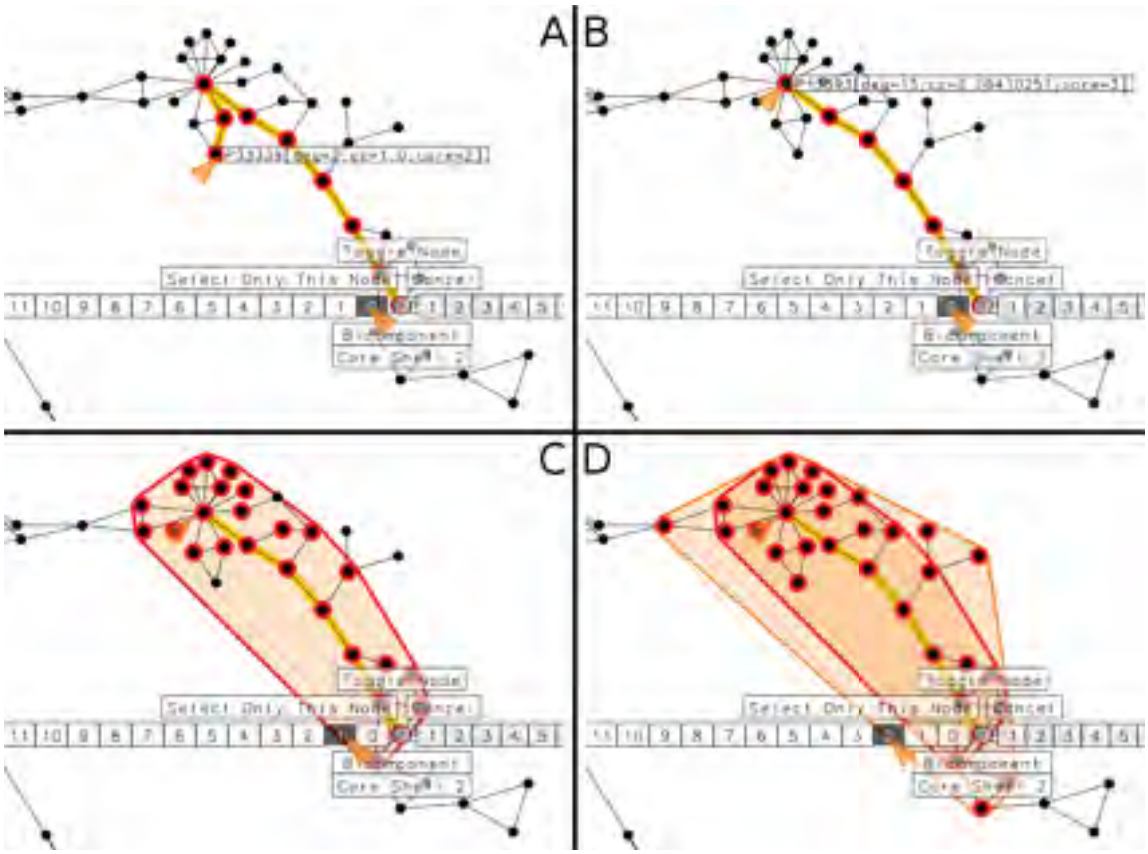


Figure 2.11 La suite de la figure précédente. A : Sélection du plus court chemin entre le premier nœud et le nœud en dessous de la MND. B : Un déplacement de la MND met à jour le chemin. C : Sélection des voisins du plus court chemin, à une distance de 1 arête. D : Sélection des voisins du plus court chemin, à une distance de 2 arêtes.

Le menu est lancé sur un nœud quelconque, et permet de sélectionner juste le nœud, ou bien le voisinage des nœuds à l'intérieur d'une distance donnée (en glissant à droite dans le menu), ou bien un sous-graphe pré-calculé contenant le nœud (en glissant vers le bas), ou bien le plus court chemin entre le nœud de départ et le nœud de la MND. Dans le dernier cas, l'utilisateur peut aussi glisser vers la gauche avec leur MD pour aussi sélectionner des voisins du chemin, jusqu'á une distance donnée.

Le fait que l'utilisateur puisse faire un survol avec le curseur de la MND, pour voir quels chemins sont mis en surbrillance, permet à l'utilisateur d'explorer et analyser le graphe de manière visuelle. Par exemple, en gardant un nœud fixe et en promenant l'autre curseur, l'utilisateur peut voir si le chemin est souvent long ou court, ou bien voir par quels nœuds intermédiaires le chemin passe.

Une autre façon d'utiliser les deux pointeurs, que nous n'avons pas implementée, serait d'utiliser les deux mains pour sélectionner un voisinage de nœuds. Imaginez un voisinage de rayon $R$ arêtes, centré sur un nœud $n_1$. L'utilisateur pourrait choisir $n_1$ avec leur MD, et placer leur MND pour choisir un autre nœud $n_2$ situé sur la périphérie du voisinage. La distance entre $n_1$ et $n_2$ donnerait $R$, et le système pourrait mettre en surbrillance le voisinage de rayon $R$ autour de $n_1$. En éloignant ou en rapprochant les deux nœuds, le voisinage serait élargi ou rapetissé. Ceci permettrait à l'utilisateur de visualiser tous les nœuds qui sont au moins aussi près de $n_1$ que $n_2$.

### 2.3.4 Technique bimanuelle de parcours de chemin

Les résultats des expériences de Chen *et al.* (2008) semblent montrer que, malgré les avantages du ToolGlass, l'utilisateur peut être ralentit s'il a besoin de porter son attention à deux retours visuels (position de la palette et position du curseur) en même temps qu'il coordonne les mouvements de ses deux mains. Il existe d'autres exemples d'interfaces bimanuelles qui semblent propices à éviter ce problème, par exemple, l'interface 3D de Balakrishnan et Kurtenbach (1999) que nous avons vue dans la section 1.3.3.2. Dans leur interface, la MND manipule le point de vue d'une caméra 3D, pendant que la MD manipule le contenu 3D. Intuitivement, ce genre d'interaction ne devrait pas obliger l'utilisateur à partager leur attention visuelle entre deux stimuli, car il y a seulement un curseur (de la MD) à surveiller, et les mouvements de la MND ne font que translater, zoomer, ou tourner le point de vue, ce qui est visible dans tout le champ visuel de l'utilisateur. Nous avons donc cherché une approche analogue pour la visualisation des réseaux. L'idée présentée dans cette section est d'utiliser la MND pour contrôler la vue de caméra 2D, pendant que la MD sélectionne des nœuds.

Plus particulièrement, l'interface proposée dans cette section permet de faire un parcours de chemin (suite de nœuds). Dans la littérature antérieure, des interfaces ont déjà été proposées pour parcourir un chemin (Moscovich *et al.*, 2009), mais en utilisant une seule main. Dans le travail de Moscovich *et al.* (2009), un graphe des vols d'avions entre des aéroports internationaux est affiché par-dessus une carte du monde, et l'utilisateur doit parcourir d'un aéroport à un autre, suivant les vols, parfois faisant un petit saut géographique à l'intérieur du même pays, et parfois en faisant un long saut pour traverser un océan. Pour faciliter cette tâche, Moscovich *et al.* (2009) proposent deux techniques unimanuelles pour plus facilement voir les voisins d'un aéroport donné, indépendemment de l'échelle géographique. Avec l'interface bimanuelle de cette section, nous sommes en mesure de proposer une solution alternative.

Figure 2.12 montre un scénario où l'utilisateur traverse un chemin entre des aéroports dans l'Amérique du Sud. La MND contrôle le zoom de la caméra, et la MD sélectionne les nœuds le long d'un chemin qui sera navigué. Les données d'exemple de la Figure 2.12 sont les mêmes que celles utilisées dans (Moscovich *et al.*, 2009), et notre technique de parcours de chemin bimanuelle a l'avantage de permettre des actions de zoom et de sélection simultanées, sans changement de modes.

Dans l'interface actuelle, le curseur de la MD peut pointer et cliquer sur les nœuds, en les ajoutant au chemin dessiné en jaune. À chaque fois qu'un nouveau nœud est sélectionné avec la MD, il est rajouté au chemin, et une translation automatiquement de la vue est provoquée, pour centrer le nouveau nœud. Si l'utilisateur clique sur un nœud qui n'est pas un voisin du dernier nœud sélectionné, le plus court chemin entre les deux est automatiquement sélectionné. L'utilisateur peut également conserver le curseur de la MD dans le centre de la vue, et simplement tourner le dispositif de pointage pour indiquer la direction de l'arête vers laquelle il veut ajouter un autre voisin. Pendant ce temps, la MND peut zoomer et dézoomer, portant les nœuds voisins plus ou moins loin, analogue à l'effet de Bring & Go (Moscovich *et al.*, 2009). (Notez qu'il n'y a pas de curseur pour la MND.)

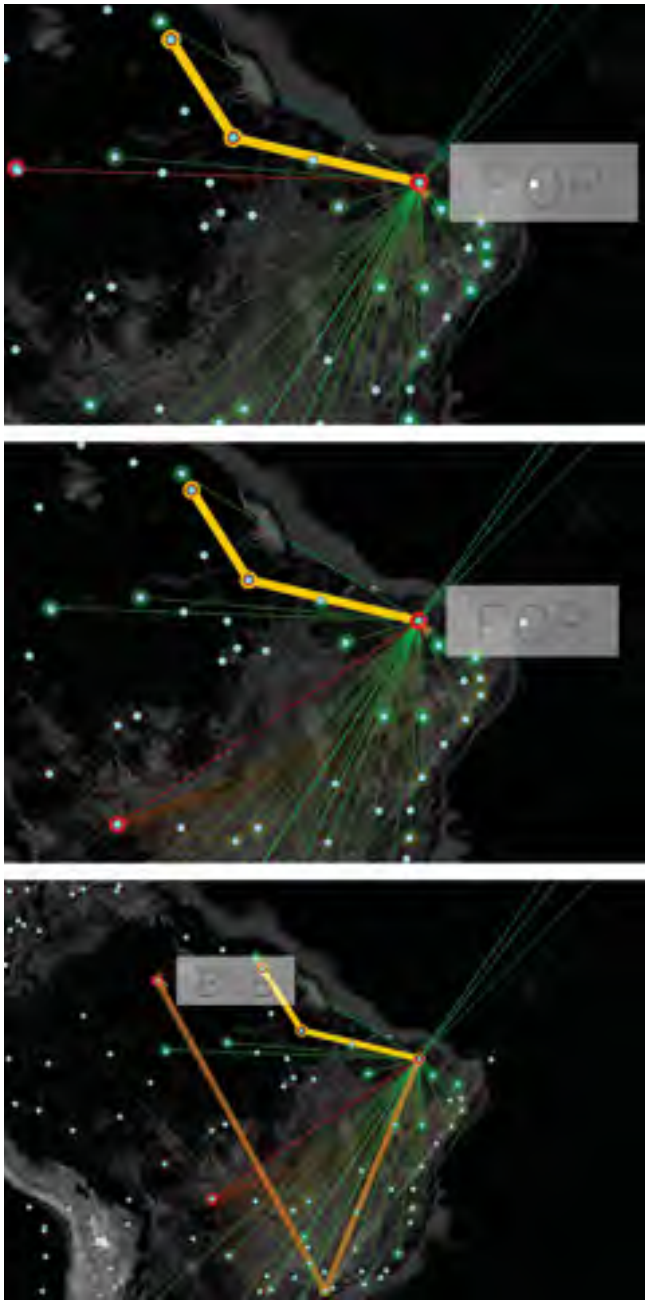

Figure 2.12 Un chemin passant par trois nœuds est sélectionné et surligné en jaune. Le curseur est actuellement sur le nœud “FOR”. À partir d’ici, l’utilisateur peut cliquer sur le prochain nœud à rajouter au chemin, ou bien tourner le dispositif de MD pour choisir la direction vers le prochain nœud. Le prochain nœud est montré par un lien en rouge vif, et ses voisins sont montrés par des liens supplémentaires en rouge plus pâle. Au milieu : Une rotation du dispositif a sélectionné un prochain voisin différent. Ci-dessous : La MND a fait un zoom arrière (*zoom out*), et la MD est maintenant par dessus un autre nœud (“BVB”), ce qui a déclenché un calcul automatique du plus court chemin jusqu’à l’ancien nœud “FOR”. Un clic à présent rajouterait tous les nœuds du plus court chemin au chemin antérieur.

### 2.3.5 La lentille MultiVisu

Les sections 1.4.2-1.4.4 ont montré que les graphes peuvent être représentés sous au moins 3 formes différentes (nœuds-liens, matricielle, et nuages de points) et même des mélanges de formes (comme dans le cas de NodeTrix, Figure 1.13). Pour changer la forme de la représentation d'un graphe ou d'un sous-graphe, une approche serait de permettre à l'utilisateur de sélectionner les nœuds désirés, et ensuite de lancer une commande dans le HotGlass pour changer la représentation de ces nœuds.

Nous avons toutefois décidé d'investiguer une approche différente pour modifier la représentation de sous-graphes. La section précédente a montré que la MND peut servir à changer la vue. En interprétant un "changement de vue" de façon plus abstraite, par analogie, la MND pourrait aussi servir à basculer entre les représentations nœuds-liens, matricielle, et en nuages de points. En pensant aussi aux lentilles qui ont déjà été proposées pour la MND (Bier *et al.*, 1993), nous sommes arrivés à l'idée d'une "lentille MultiVisu" qui servirait à choisir la représentation visuelle pour un sous-ensemble de nœuds.

Une autre source d'inspiration a été Hinckley *et al.* (2010), qui ont montré que la MND peut servir à établir un contexte transitoire dans lequel la MD travaille. Poussant ces idées plus loin, il pourrait être utile si l'utilisateur avait une façon rapide et transitoire pour modifier la représentation d'un sous-graphe, peut-être pour effectuer quelques opérations rapides avec la MD, puis revenir à la représentation de départ. La lentille MultiVisu Lens est proposée pour implémenter cette idée (Figure 2.13).

La lentille peut être invoquée sur un ensemble pour changer sa représentation visuelle. Lorsque le bouton MND.1 est appuyé sur un ensemble, la lentille MultiVisu apparaît, et elle reste en place aussi longtemps que le bouton MND.1 est appuyé. En même temps, le curseur de la MND peut faire un mouvement de glissement dans un panneau de boutons à gauche de la lentille et sélectionner une représentation pour les nœuds de l'ensemble, tandis que le curseur de la MD peut interagir avec la lentille. Une telle lentille pourrait être utilisée transitoirement pour examiner un ensemble dense de nœuds, pour y trouver le nœud avec le plus haut degré

(chose facile à trouver dans une représentation en nuage de points), ou bien pour vérifier si une grappe (*"cluster"*) de nœuds est vraiment une clique, ou si au contraire il manque quelques arêtes (chose facile à voir dans une représentation matricielle).

Un exemple de suite d'opérations est montrées dans la Figure 2.13. D'abord, l'utilisateur effectue une sélection en lasso (Figure 2.13 A) pour définir un ensemble (B). Ensuite, la MND fait afficher une lentille MultVisu sur l'ensemble, montrant le contenu de la lentille sous forme nœuds-liens (C), ensuite sous forme matricielle (D, E), ensuite en nuages de points (F). Ensuite, la MND relâche son bouton au dessus d'un icône de punaise pour garder la lentille affichée (G) et pour libérer le curseur de la MND. Finalement, la MD ferme le panneau d'options de la lentille (H).

Dans le prototype actuel, trois métriques sont calculées sur chaque nœud : le degré ("deg"), le coefficient de regroupement ou *clustering coefficient* ("cc"), et la couche de décomposition $K$-noyaux ("core"). Le panneau situé à gauche de la lentille comprend une moitié triangulaire d'un SPLOM ayant trois nuages de points, formés par ces trois métriques. Dans le même panneau, on retrouve les représentations nœuds-liens ("N-L") et matricielle ("MAT"). La MND peut choisir n'importe quelle de ces représentations dans le panneau à gauche. Si la représentation nœuds-liens est choisie, alors une barre de défilement apparaît sur le côté droit de la lentille, permettant à la MD de varier l'opacité de nœuds en fonction de leur degré (Figure 2.13 C). (Cette barre interpole entre deux états : à l'extrémité supérieure, tous les nœuds sont dessinés avec une opacité complète, et à l'autre extrémité, l'opacité de chaque nœud est proportionnelle à son degré, pour mettre en évidence les nœuds de degré élevé. Ces nœuds seraient peut-être cachés par leurs voisins autrement.) Si la représentation matricielle est choisie, les noms des nœuds apparaissent sur le côté droit de la lentille, et l'utilisateur peut glisser-déposer ces noms avec la MD (Figure 2.13 D) pour réordonnancer la matrice. La MD peut aussi cliquer sur une option "Reorder" pour lancer un algorithme de réordonnancement automatique de la matrice

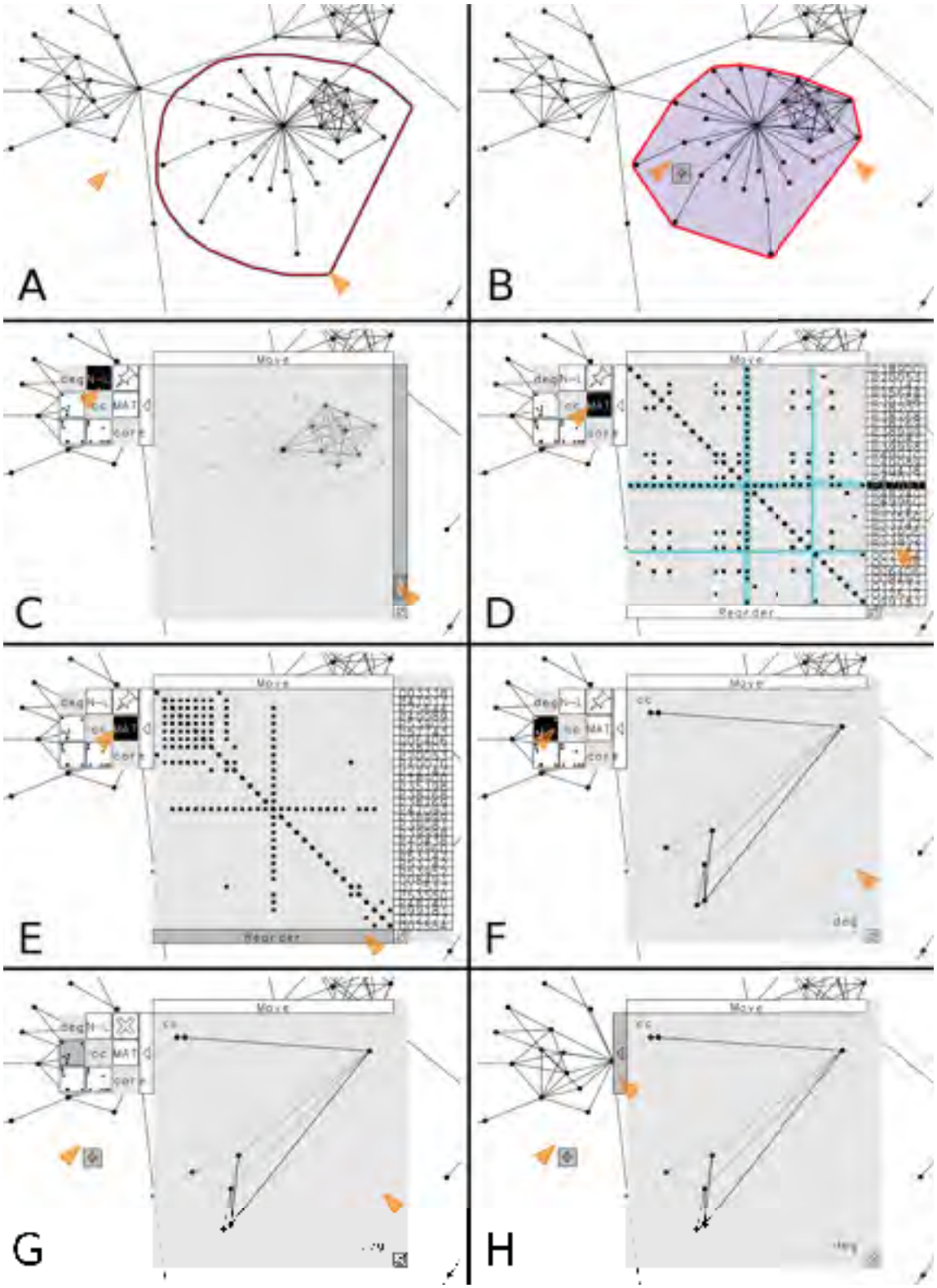


Figure 2.13 Création de, et interactions avec, une lentille MultiVisu.

(Figure 2.13 E), pour mettre en évidence des grappes de nœuds qui seront “attirés” vers la diagonale par le réordonnancement.

Tant et aussi longtemps que la MND garde le bouton MND.1 enfoncé, la lentille reste en place. En relâchant le bouton MND.1, la lentille disparaît et l’ensemble de nœuds revient à sa représentation nœuds-liens normale, permettant donc une opération transitoire. Donc, si l’utilisateur voit une grappe de nœuds sous forme nœuds-liens, et veut simplement sélectionner le nœud avec le degré le plus élevé dans la grappe, il pourra sélectionner la grappe et en définir un ensemble, faire apparaître une lentille MultiVisu sur l’ensemble, basculer vers un nuage de points avec le degré sur un des axes, sélectionner le ou les nœuds avec le degré plus élevé, et lâcher le bouton de la MND pour revenir à une représentation nœuds-liens, comme si la lentille n’était qu’un widget contextuel. Toutefois, si l’utilisateur veut garder la lentille à l’écran, il peut relâcher le curseur de la MND sur l’icône de punaise. Dans ce cas, la lentille reste affichée à l’écran (Figure 2.13 G), un peu comme un *tear-off menu*, ou comme un FaST Slider (McGuffin *et al.*, 2002). Ensuite, la lentille peut être déplacée, redimensionnée, ou éliminée en cliquant sur l’icône “X”, ou encore d’autres lentilles peuvent être invoquées ailleurs (Figure 2.14), ce qui donne une visualisation de réseau hybride similaire à NodeTrix (Henry *et al.*, 2007), mais qui permet des vues de nuage de points ainsi que des vues matricielles. La visualisation qui en résulte est donc une extension plus flexible de NodeTrix.

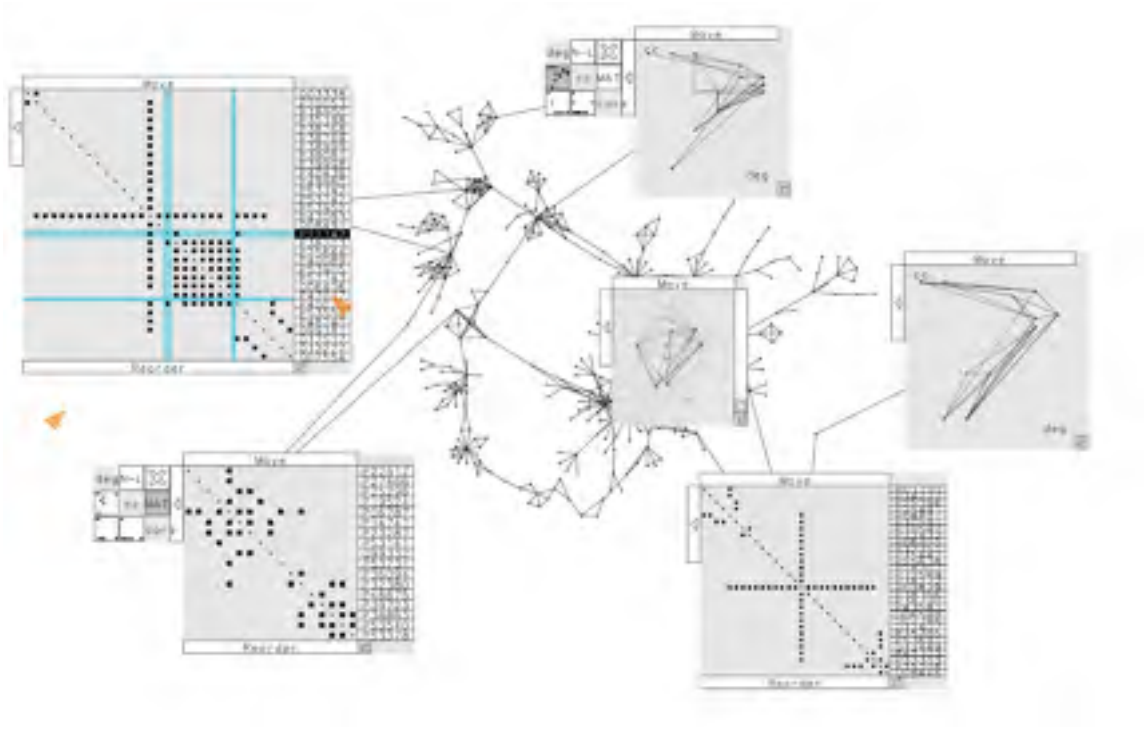

Figure 2.14 Un graphe avec plusieurs lentilles MultiVisu, montrant différentes parties du graphe sous forme nœuds-liens, matricielle, et en nuages de points. Le résultat est une visualisation hybride du graphe.

# CHAPITRE 3

# PREMIÈRES RÉACTIONS D'UTILISATEURS

Pour solliciter des premières réactions d'utilisateurs, le prototype fût montré à 10 utilisateurs : 5 membres d'un laboratoire de bio-informatique (2 programmeurs, 1 doctorant, 1 post doctorant, 1 associé scientifique), et 5 étudiants en génie (1 de premier cycle, 3 étudiants en maîtrise, 1 doctorant). Ils avaient tous au moins un peu d'expérience à visualiser et à travailler avec des données en réseau. Huit sont droitiers, et les deux autres ont dit qu'ils utilisent la souris avec la main droite, donc tous les dix ont utilisé le prototype avec la main droite jouant le rôle de la MD. Chaque utilisateur a passé environ 1 heure avec le logiciel, pendant laquelle l'interface a été montrée à l'utilisateur, et l'utilisateur a eu à effectuer quelques tâches de manière informelle.

## 3.1 Manipulations bimanuelles symétriques versus asymétriques

D'abord, les utilisateurs ont eu à essayer des manipulations bimanuelles symétriques et asymétriques, pour voir lesquelles ils préfèrent. Tel qu'expliqué dans la section 2.3.1, notre prototype permet des manipulations de caméra bimanuelles symétriques ou asymétriques, et permet aussi une manipulation directe (translation, rotation, changement d'échelle) bimanuelle d'un ensemble de nœuds qui, elle aussi, peut être symétrique ou asymétrique. Les utilisateurs ont eu à effectuer quelques opérations de caméra (zoomer sur un ensemble, ensuite dézoomer pour voir tout le graphe, avec une translation à chaque fois, répété à quelques reprises) en mode symétrique et ensuite asymétrique. Ils ont eu aussi à saisir quelques ensembles avec les deux mains et les tourner et déplacer à quelques reprises, d'abord en mode symétrique et ensuite asymétrique. Les utilisateurs ont eu à faire chacune de ces tâches 1, 2, ou 3 fois, jusqu'à ce qu'ils développent une préférence pour une interaction symétrique ou asymétrique. La plupart des utilisateurs ont préféré une interaction asymétrique :

“Quel type de manipulation bimanuelle préférez-vous avec la caméra ?”

Asymétrique : (7)
Symétrique : (1)
Sans préférence : (2)

“Quel type de manipulation bimanuelle préférez-vous pour translater, tourner, et changer l’échelle d’un ensemble ?”

Asymétrique : (7)
Symétrique : (2)
Sans préférence : (1)

Cette préférence peut sembler surprenante, car dans les interfaces multitactiles, le contrôle de caméra et la manipulation directe se fait habituellement de façon symétrique, et les interactions symétriques sont aussi plus faciles à comprendre par un nouvel utilisateur. Toutefois, pour des dispositifs indirects comme ceux de notre prototype, nous trouvons qu’il est plus facile de contrôler la translation, rotation, et changement d’échelle de façon indépendante avec une interaction asymétrique, et cela semble être la raison derrière les choix des utilisateurs.

## 3.2 HotGlass : tâches MC versus tâches MCMO

Tel qu’expliqué dans la section 2.3.2, nous soupçonnons qu’une interaction clic-à-travers, comme avec un ToolGlass, est mieux pour des tâches MCMO (car la commande et l’objet peuvent être sélectionnés en même temps), et une interaction de genre HotBox (tel que permis en téléportant le HotGlass) est mieux pour des tâches MC, car l’utilisateur a seulement besoin de sélectionner l’objet une fois et peut ensuite faire afficher le widget et sélectionner une suite de commandes. (Les tâches MO, pour leur part, sont exécutables avec des outils modaux qui peuvent être mis à l’intérieur soit d’un ToolGlass ou d’un HotBox).

Pour investiguer si les utilisateurs auraient une préférence pour une ou l’autre façon d’interagir, ils ont eu à effectuer quatre séries de tâches avec le HotGlass, couvrant les combinaisons { MCMO, MC } $\times$ { clic-à-travers comme un ToolGlass, téléportation comme un HotBox }.

Pour chaque série de tâches MCMO, il y avait 4 ensembles de nœuds présentés à l’utilisateur, et l’utilisateur devait changer la forme des nœuds dans le premier ensemble, faire un “Flip”

complet de l'ensemble, ensuite changer la forme des nœuds du deuxième ensemble, faire un "Flip" complet de ce deuxième ensemble, etc. Par exemple, on pouvait demander à l'utilisateur de changer la forme des nœuds du premier ensemble en cercles, du deuxième en triangles, etc. La commande de "Flip" entre les changements de forme était pour obliger l'utilisateur à déplacer la MD d'une commande à l'autre, quelque soit la technique d'interaction utilisée. Les utilisateurs faisaient une série de tâches en cliquant-à-travers, ensuite une série semblable en téléportant. On s'attendait à ce que la téléportation semble moins conviviale pour les tâches MCMO, car elle ne permet pas de sélectionner l'objet et la commande en même temps, et oblige plutôt à l'utilisateur de basculer constamment entre la sélection de l'objet et l'affichage du menu.

Pour chaque série de tâches MC, il y avait un seul ensemble présenté à l'utilisateur, et l'utilisateur devait changer la forme des nœuds de l'ensemble, faire un "Flip" complet de l'ensemble, ensuite changer encore la forme des mêmes nœuds, faire un autre "Flip", etc. Par exemple, on pouvait demander à l'utilisateur de changer la forme des nœuds en cercles, ensuite en triangles, etc. Encore une fois, les utilisateurs faisaient une série de tâches en cliquant-à-travers, ensuite une série semblable en téléportant. Cette fois ci, on s'attendait à ce que l'interaction par clic-à-travers semble moins conviviale pour des tâches MC, car elle oblige l'utilisateur de coordonner les deux mains pour chaque commande, contrairement à la téléportation qui demanderait juste de sélectionner l'ensemble et afficher le menu une seule fois pour toute la série.

"Pour la réalisation d'une tâche MCMO, quel paradigme préférez-vous ?"

Cliquer-à-travers (comme un ToolGlass) : ■■■■■■■ (7)
Téléporter (comme un HotBox) : ■■ (2)
Sans préférence : ■ (1)

"Pour la réalisation d'une tâche MC, quel paradigme préférez-vous ?"

Cliquer-à-travers (comme un ToolGlass) : ■■ (2)
Téléporter (comme un HotBox) : ■■■■■■■■ (8)

Les résultats s'alignent avec nos attentes. Notez aussi que, dans le cas de la tâche MC, les deux utilisateurs qui ont préféré l'interaction en cliquant-à-travers ont dit que la téléportation

semblait moins “intuitive”, mais un de ces utilisateurs a quand même dit que la téléportation semblait plus rapide.

### 3.3 Réactions générales face aux dispositifs bimanuels

Suite aux tâches, les utilisateurs ont été interrogés pour savoir s’ils aimeraient utiliser, ou essayer d’utiliser une deuxième souris si elle était disponible pour eux dans des applications réelles.

“Aimeriez-vous avoir deux souris connectées à votre ordinateur si certaines de vos applications permettaient des interactions bimanuelles ?”

Oui : ■■■■■ (5)
Non : ■ (1)
Ça dépend : ■■■■ (4)

4 utilisateurs ont donné des réponses nuancées en disant que cela dépend d’autres facteurs. Les utilisateurs ont déclaré que l’intérêt d’utiliser une deuxième souris dépend de la tâche à accomplir, et 1 utilisateur a également exprimé sa préoccupation au sujet de l’espace limité disponible pour une souris supplémentaire sur son bureau. Un autre utilisateur a dit “pour un utilisateur débutant, [une interface bimanuelle] me ferait peur”, mais que pour un utilisateur expert, “une fois que l’utilisateur s’y habitue, j’imagine que l’interface serait très rapide”.

### 3.4 Sélection bimanuelle du plus court chemin

Tel que expliqué dans la section 2.3.3, notre prototype ne permet pas seulement de sélectionner le plus court chemin entre deux curseurs, mais recalcule ce chemin et le met en surbrillance de façon dynamique, permettant de faire un survol du curseur de la MND pour explorer différents chemins. Nous avons montré cette fonctionnalité au directeur d’un laboratoire de bioinformatique qui travaille souvent avec des données en graphe. Il nous a expliqué que cette mise en surbrillance dynamique du plus court chemin pourrait être utile dans l’examen de réseaux biologiques, par exemple, les réseaux d’interaction protéine-protéine et les réseaux bipartis microARN-cible. Par exemple, dans les réseaux d’interaction protéine-protéine, certaines des

protéines clés pourraient ne pas avoir un degré élevé, mais elles pourraient être quand même impliquées dans de nombreux plus courts chemins. Le directeur a aussi affirmé que, bien sûr, ces nœuds pourraient être retrouvés analytiquement si l'utilisateur fait une requête explicite, mais les outils interactifs ou visuels peuvent permettre à l'utilisateur d'examiner d'autres cibles de façon dynamique et de faire des découvertes inespérées.

# CONCLUSION

Quatre nouvelles techniques bimanuelles ont été présentées pour interagir avec la visualisation d'un réseau :

- Une technique bimanuelle pour la sélection du plus court chemin et ses voisins, intégrée à l'intérieur d'un menu contextuel qui permet d'autres sélections (voisinage, et sous-graphes pré-calculés) et qui est extensible.
- HotGlass, qui est une fusion novatrice du ToolGlass et du HotBox, et qui combine des avantages des deux. HotGlass est conçu pour permettre une exécution rapide de tâches MC, MO, et MCMO. Dans le cas de tâches MC, l'objet a seulement besoin d'être sélectionné une fois ; dans le cas de tâches MO, un outil modal aurait besoin d'être sélectionné une seule fois ; et dans le cas de tâches MCMO, chaque couple (objet, commande) peut être sélectionné en même temps par un clic-à-travers. En ce sens, le HotGlass est unique par rapport aux techniques antérieures.
- Deux nouvelles techniques d'interaction bimanuelles inspirées en pensant par analogie avec des techniques existantes :
  - Une façon bimanuelle de parcourir un chemin, où la MND contrôle la caméra.
  - La lentille MultiVisu.

Nous avons donc atteint les objectifs de la section 1.6.

Concernant le HotGlass, notons que cette technique d'interaction n'est pas limitée aux visualisations de graphes, mais pourrait s'appliquer plus généralement dans des interfaces qui ne sont même pas des visualisations, par exemple dans des interfaces de dessin (comme l'application de Kurtenbach *et al.* (1997)), ou dans n'importe quelle application graphique où l'utilisateur doit sélectionner des commandes et des objets.

Concernant la lentille MultiVisu, elle se distingue des autres widgets contextuels pour la visualisation (Moscovich *et al.*, 2009; McGuffin et Jurisica, 2009; Bezerianos *et al.*, 2010b; Viau

*et al.*, 2010) dans le sens qu'elle est opérée en utilisant deux dispositifs de pointage. Peut-être encore plus intéressant, l'utilisateur peut créer plusieurs instances de lentilles MultiVisu pour créer une visualisation hybride du graphe qui mélange les représentations nœuds-liens, matricielle, et en nuages de points, ce qui étend l'hybride de NodeTrix (Henry *et al.*, 2007) qui est limité aux deux premières représentations.

Nous avons aussi présenté une taxonomie novatrice de techniques d'interaction bimanuelles pour les visualisations (section 2.1) qui classe des techniques existantes et nos nouvelles techniques. À l'avenir, cette taxonomie pourrait servir pas juste pour les visualisations de graphes, mais pour les visualisations de données en générale.

Nous avons aussi présenté des commentaires d'utilisateurs qui, tel que nous nous attendions, montrent une préférence pour l'interaction clic-à-travers pour les tâches MCMO, et la téléportation pour les tâches MC, ce qui justifie en partie la conception du HotGlass. D'autres commentaires présentés (section 3.4) justifient en partie la conception de notre technique de sélection du plus court chemin.

Passons maintenant aux limitations de ces contributions.

Dans le cas de notre technique de sélection du plus court chemin, nous ne savons pas encore s'il y a une utilité à permettre de sélectionner les *voisins* du chemin. Aussi, cette technique se trouve à l'intérieur d'un menu extensible, permettant la sélection d'autres sous-ensembles, mais ceci empêche d'explorer des plus courts chemins en changeant les *deux* extrémités du chemin en même temps, avec des glissements des deux curseurs. Pour déterminer quel genre de fonctionnalité est plus utile, il faut consulter avec des utilisateurs et analyser leurs tâches avec les graphes.

Dans le cas de HotGlass, cette technique combine en théorie les avantages de ToolGlass et de HotBox, et permet d'accéder à des milliers de commandes (comme le HotBox (Kurtenbach *et al.*, 1999)). Toutefois, nous savons aussi que ToolGlass est plus lent que les Control Menus (Guimbretière *et al.*, 2005; Chen *et al.*, 2008) pour des tâches simples quand seulement un petit

nombre de commandes est nécessaire. De plus, nous avons remarqué subjectivement que, avec HotGlass, il semble y avoir un coût supplémentaire par rapport à ToolGlass ou HotBox, parce qu'à chaque fois que l'utilisateur veut lancer une commande avec HotGlass, il doit prendre une décision : est-il mieux de cliquer-à-travers, ou bien de téléporter ? Il se pourrait que, avec l'habitude, cette décision devienne plus facile et rapide. Mais seulement une expérience contrôlée pourra comparer les temps nécessaires entre HotGlass, ToolGlass pur, HotBox pur, et d'autres techniques, pour voir laquelle est plus rapide. Il se pourrait que Control Menus se montre toujours plus rapide que ces autres techniques, mais seulement pour un petit nombre de commandes. Si c'est le cas, une interface encore plus poussée pourrait mettre les commandes les plus fréquentes dans un Control Menu pour un accès rapide, et mettre les autres commandes dans un HotGlass, qui est capable de stocker un grand nombre de commandes.

Dans le cas de la technique de parcours de chemin bimanuel, nous ne savons pas si cette technique est plus rapide qu'une technique comparable unimanuelle comme Bring & Go (Moscovich *et al.*, 2009). Seulement une expérience contrôlée pourra le déterminer.

Dans le cas de la lentille MultiVisu, nous ne savons pas si cette technique bimanuelle a un avantage véritable par rapport à une technique équivalente unimanuelle. (Notons toutefois qu'aucune technique unimanuelle a été proposée, jusque'à maintenant, pour basculer entre les trois représentations permises par notre lentille MultiVisu.) Encore une fois, une expérience contrôlée serait nécessaire pour répondre à cette question.

Pour les futurs travaux, il serait intéressant d'itérer sur la conception de la technique de sélection de plus courts chemins et de sous-graphes, et d'évaluer les différentes techniques bimanuelles pour décortiquer leurs avantages ou inconvénients en termes de performance. Une autre direction future intéressante serait de voir à quel point les idées dans ce mémoire pourraient s'appliquer à la conception d'interfaces multitactiles.

## BIBLIOGRAPHIE


Apitz, G. et F. Guimbretière. 2004. CrossY : a crossing-based drawing application. *In Proc. ACM Symposium on User Interface Software and Technology (UIST)*, pages 3–12.

Balakrishnan, R. et G. Kurtenbach. 1999. Exploring bimanual camera control and object manipulation in 3D graphics interfaces. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 56–62.

Balakrishnan, R., T. Baudel, G. Kurtenbach, et G. Fitzmaurice. 1997. The Rockin'Mouse : integral 3D manipulation on a plane. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 311–318.

Beaudouin-Lafon, M., W.E. Mackay, P. Andersen, P. Janecek, M. Jensen, M. Lassen, K. Lund, K. Mortensen, S. Munck, A. Ratzer, K. Ravn, S. Christensen, et K. Jensen. 2001. CPN/-Tools : A post-WIMP interface for editing and simulating coloured petri nets. *In Proc. Int. Conf. on Application and Theory of Petri Nets (ICATPN)*, pages 71–80.

Bezerianos, A., F. Chevalier, P. Dragicevic, N. Elmqvist, et J.D. Fekete. 2010a. « GraphDice : A system for exploring multivariate social networks ». *Computer Graphics Forum (Proceedings of EuroVis)*, pages 863–872.

Bezerianos, A., P. Dragicevic, J.-D. Fekete, J. Bae, et B. Watson. 2010b. « GeneaQuilts : A System for Exploring Large Genealogies ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 16(6) :1073–1081.

Bier, E. A., Maureen C. Stone, Ken Pier, William Buxton, et Tony D. DeRose. 1993. Toolglass and magic lenses : The see-through interface. *In Proc. ACM SIGGRAPH*, pages 73–80.

Buxton, W. 1983. « Lexical and pragmatic considerations of input structures ». *Computer Graphics*, 17(1) :31–37.

Buxton, W. 1990. A three-state model of graphical input. *In Proc. INTERAC Conference on Human-Computer Interaction*, pages 449–456.

Buxton, W. 1986. Chunking and phrasing and the design of human-computer dialogues. *In Proc. IFIP World Computer Congress*, pages 475–480.

Callahan, J., D. Hopkins, M. Weisert, et B. Shneiderman. 1988. An empirical comparison of pie vs. linear menus. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 95–100.

Chen, N., F. Guimbretière, et C.E. Löckenhoff. 2008. « Relative role of merging and two-handed operation on command selection speed ». *International Journal of Human-Computer Studies*, 66 :729–740.

Di Battista, G., P. Eades, R. Tamassia, et I. G. Tollis, 1999. *Graph Drawing : Algorithms for the Visualization of Graphs*. Prentice-Hall.

Dwyer, T., B. Lee, D. Fisher, K.I. Quinn, P. Isenberg, G. Robertson, et C. North. 2009. « A comparison of user-generated and automatic graph layouts ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, pages 961–968.

Frisch, M., J. Heydekorn, et R. Dachselt. 2009. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. *In Proc. ACM Conference on Interactive Tabletops and Surfaces (ITS)*, pages 149–156.

Ghoniem, M., J.D. Fekete, et P. Castagliola. 2004a. Comparaison de la lisibilité des graphes en représentation nœuds-liens et matricielle. *In Proc. AFIHM Conférence de l'Association Francophone d'Interaction Homme-Machine*, pages 77–84.

Ghoniem, M., J.D. Fekete, et P. Castagliola. 2004b. A comparison of the readability of graphs using node-link and matrix-based representations. *In Proc. IEEE Symposium on Information Visualization (InfoVis)*, pages 17–24.

Grossman, T. et Ravin Balakrishnan. 2005. The bubble cursor : Enhancing target acquisition by dynamic resizing of the cursor's activation area. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 281–290.

Guiard, Y. 1987. « Asymmetric division of labor in human skilled bimanual action : The kinematic chain as a model ». *Journal of Motor Behavior*, 19 :486–517.

Guimbretière, F. et T. Winograd. 2000. FlowMenu : combining command, text, and data entry. *In Proc. ACM Symposium on User Interface Software and Technology (UIST)*, pages 213–216.

Guimbretière, F., A. Martin, et T. Winograd. 2005. « Benefits of merging command selection and direct manipulation ». *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(3) :460–476.

Hancock, M. S., Frederic D. Vernier, Daniel Wigdor, Sheelagh Carpendale, et Chia Shen. 2006. Rotation and translation mechanisms for tabletop interaction. *IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop)*, pages 79–88.

Henry, N., J.D. Fekete, et M.J. McGuffin. 2007. « NodeTrix : a hybrid visualization of social networks ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, pages 1302–1309.

Hinckley, K., Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, et Bill Buxton. 2010. Manual deskterity : an exploration of simultaneous pen + touch direct input. *In Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 2793–2802.

Holten, D. et J.J. Van Wijk. 2009. « Force-Directed Edge Bundling for Graph Visualization ». *Computer Graphics Forum (Proceedings of EuroVis)*, pages 983–990.

Kabbash, P., W. Buxton, et A. Sellen. 1994. Two-handed input in a compound task. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 417–423.

Kaufmann, M. et Dorothea Wagner, editors, 2001. *Drawing Graphs : Methods and Models*. Springer.

Kurtenbach, G. et W. Buxton. 1994. User learning and performance with marking menus. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 258–264.

Kurtenbach, G., G.W. Fitzmaurice, R.N. Owen, et T. Baudel. 1999. The hotbox : efficient access to a large number of menu-items. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 231–237.

Kurtenbach, G. et William Buxton. 1993. The limits of expert performance using hierarchic marking menus. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 482–487.

Kurtenbach, G., George Fitzmaurice, Thomas Baudel, et Bill Buxton. 1997. The design of a GUI paradigm based on tablets, two-hands, and transparency. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 35–42.

Kurtenbach, G. P. 1993. *The Design and Evaluation of Marking Menus*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada.

Latulipe, C., S. Mann, C.S. Kaplan, et C.L.A. Clarke. 2006. symSpline : symmetric two-handed spline manipulation. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 349–358.

Lee, B., C. Plaisant, C.S. Parr, J.D. Fekete, et N. Henry. 2006. Task taxonomy for graph visualization. *In Proc. AVI workshop on BEyond time and errors : novel evaLuation methods for Information Visualization (BELIV)*, pages 1–5.

Leganchuk, A., Shumin Zhai, et William Buxton. 1998. « Manual and cognitive benefits of two-handed input : an experimental study ». *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5 :326–359.

McGuffin, M., Nicolas Burtnyk, et Gordon Kurtenbach. 2002. FaST sliders : Integrating Marking Menus and the adjustment of continuous values. *In Proc. Graphics Interface (GI)*, pages 35–41.

McGuffin, M. J. et Igor Jurisica. 2009. « Interaction Techniques for Selecting and Manipulating Subgraphs in Network Visualizations ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(6) :937–944.

Moscovich, T., Fanny Chevalier, Nathalie Henry, Emmanuel Pietriga, et Jean-Daniel Fekete. 2009. Topology-aware navigation in large networks. *In Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 2319–2328.

Norman, D., 2002. *The Design of Everyday Things*. Basic books.

Otjacques, B. et F. Feltz. 2005. Representation of graphs on a matrix layout. *International Conference on Information Visualisation (IV)*, pages 339–344.

Owen, R., G. Kurtenbach, G. Fitzmaurice, T. Baudel, et B. Buxton. 2005. When it gets more difficult, use both hands : exploring bimanual curve manipulation. *In Proc. Graphics Interface (GI)*, pages 17–24.

Pook, S., Eric Lecolinet, Guy Vaysseix, et Emmanuel Barillot. 2000. Control menus : Excecution and control in a single interactor. *In Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 263–264.

Saund, E., David Fleet, Daniel Larner, et James Mahoney. 2003. Perceptually-supported image editing of text and graphics. *In Proc. ACM Symposium on User Interface Software and Technology (UIST)*, pages 183–192.

Schmidt, S., M.A. Nacenta, R. Dachselt, et S. Carpendale. 2010. A set of multi-touch graph interaction techniques. *In Proc. ACM Conference on Interactive Tabletops and Surfaces (ITS)*, pages 113–116.

Sellen, A. J., Gordon P. Kurtenbach, et William A. S. Buxton. 1992. « The prevention of mode errors through sensory feedback ». *Human Computer Interaction*, 7(2) :141–164.

Shen, Z. et K.-L. Ma. 2007. Path visualization for adjacency matrices. *In Proc. Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*, pages 83–90.

Shneiderman, B. August 1983. « Direct manipulation : a step beyond programming languages ». *IEEE Computer*, 16(8) :57–69.

Van Ham, F. 2003. Using multilevel call matrices in large software projects. *In Proc. IEEE Symposium on Information Visualization (InfoVis)*, pages 227–232.

Viau, C., M.J. McGuffin, Y. Chiricota, et I. Jurisica. 2010. « The FlowVizMenu and Parallel Scatterplot Matrix : Hybrid Multidimensional Visualizations for Network Exploration ». *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 16(6) : 1100–1108.

Wikipedia. Juin 2011. Pie menu. URL `http://en.wikipedia.org/wiki/Marking_menu`.