publié 2003 dans la Revue d'Interaction Homme-Machine **4**(1), pp 59–84

« Control menus » et vues contextuelles pour les interfaces zoomables

Stuart POOK[1,2], Eric LECOLINET[1], Guy VAYSSEIX[2,3], Emmanuel BARILLOT[2,3]

[1] École Nationale Supérieure des Télécommunications et CNRS URA 820
Département Informatique et Réseaux
46, rue Barrault
75634 Paris cedex 13, France
stuart@acm.org, elc@enst.fr

[2] Infobiogen
523, place des Terrasses de l'Agora
91034 Évry cedex, France

[3] Généthon
1 bis, rue de l'Internationale, BP 60
91002 Évry cedex, France

RÉSUMÉ : Les interfaces zoomables constituent un outil intéressant pour visualiser des espaces informationnels de grande taille. Cependant, le fait que ces interfaces fournissent généralement peu de contexte peut parfois rendre leur utilisation délicate. Les utilisateurs ont tendance à avoir du mal à se localiser précisément dans l'espace d'information et à déterminer avec certitude la nature de leur point de focus. Nous proposons deux nouvelles aides temporaires transparentes qui facilitent la contextualisation du focus courant dans l'espace d'information. Nous proposons également une représentation hiérarchique qui indique en permanence la structure de l'espace d'information et la position de l'utilisateur dans cette structure.

Nous proposons par ailleurs un nouveau type de menu contextuel, appelé « Control menu », qui permet de fluidifier l'interaction en sélectionnant et en contrôlant les opérations en un seul geste. Bien que son utilisation ne soit pas limitée à ce type d'interfaces, ce nouveau type de menu semble particulièrement bien adapté aux interfaces qui nécessitent de fréquentes interactions comme les interfaces zoomables.

Deux interfaces zoomables implémentant ces nouvelles techniques peuvent être testées à l'URL http://www.infobiogen.fr/services/zomit/.

MOTS-CLEFS : visualisation d'information, techniques d'interaction, vues focus+contexte, interfaces zoomables, transparence, menus contextuels, Control menus

1 Introduction

Les interfaces zoomables (« zoomable user interfaces » ou ZUIs) ne sont plus une nouveauté et leurs principes [Furnas et Bederson 1995] et applications pratiques [Bederson et al. 1996 ; Furnas et Zhang 1998 ; Hightower et al. 1998] ont déjà été présentés dans plusieurs publications. Quand un utilisateur interagit avec une ZUI, il voit une vue d'un espace d'information. La vue initiale montre l'espace à une échelle qui permet de l'afficher en entier sur l'écran de l'utilisateur. Celui-ci peut alors « zoomer » (agrandir) la partie de la vue qu'il trouve intéressante. Les objets graphiques s'agrandissent jusqu'à ce qu'il y ait suffisamment de place sur l'écran pour remplacer ces objets graphiques par des

représentations alternatives montrant les données sous-jacentes avec plus de détails. Nous avons employé cette technique, nommée « zoom sémantique », pour visualiser et parcourir la base de données HuGeMap qui regroupe les principales cartes génétiques et physiques du génome humain [Pook et al. 1998]. Cette ZUI a été utilisée pour expérimenter les nouvelles techniques décrites dans cet article. Pour comprendre les exemples présentés, il suffit de savoir que la première vue montre 24 chromosomes (figure 1a), que ces chromosomes possèdent trois cartes génétiques (figure 1b), et que ces cartes consistent en des marqueurs génétiques positionnés le long d'un axe (figures 1d et 1e). Enfin, la séquence de chaque marqueur génétique lui est associée sous forme d'une chaîne de caractères.

Les ZUIs peuvent constituer un outil performant pour visualiser des espaces informationnels de grande taille. Elles permettent en particulier de parcourir l'espace d'information et d'y appliquer des transformations via des outils spécifiques comme des lentilles magiques (les lentilles magiques, qui ont été initialement développées à Xerox PARC [Stone et al. 1994], sont des fenêtres qui peuvent être créées et déplacées interactivement pour afficher le résultat d'une transformation s'appliquant à la région qu'elles recouvrent). Un des problèmes des ZUIs (et plus généralement des représentions d'espaces informationnels de grande taille) est que la vue de l'information présentée aux utilisateurs, le *focus*, ne comporte pas toujours suffisamment de *contexte* pour que ceux-ci puissent situer précisément ce qu'ils voient dans l'espace d'information. Cette situation peut provoquer des phénomènes de désorientation : les utilisateurs sont alors « perdus dans l'hyperespace », parfois jusqu'à ne plus comprendre ce qu'ils voient et ne plus savoir où aller. Nous présentons deux nouveaux outils temporaires que les utilisateurs peuvent faire apparaître et utiliser lorsqu'ils se trouvent dans cette situation. Le premier outil permet aux utilisateurs de situer le focus dans une vue globale de l'espace d'information. Le deuxième leur permet de parcourir en sens inverse le chemin qui les a amenés à se perdre. Nous proposons également une troisième aide de navigation qui est toujours visible et facilite la navigation en permanence. Cette aide montre la structure de l'espace d'information et la position de l'utilisateur dans cet espace. Elle peut aussi permettre à l'utilisateur de se déplacer plus rapidement dans l'espace d'information.

Quand il utilise une ZUI, un utilisateur peut zoomer, « dézoomer », faire défiler l'image, créer des lentilles magiques, déplacer des lentilles, retailler des lentilles, zoomer ou faire défiler un portail, etc. Les ZUIs sont donc des interfaces complexes, mais l'interaction se fait toujours au moyen de la souris, des menus, et des boutons selon le traditionnel modèle WIMP (« Window, Icon, Menu, Pointer »). Or, les utilisateurs doivent exécuter certaines de ces actions très fréquemment. Par exemple, un utilisateur zoomera jusqu'à ce que l'échelle désirée ait été obtenue ou fera défiler l'image jusqu'à ce que l'objet recherché ait été trouvé. Les utilisateurs sont également confrontés à des vues changeant rapidement où chaque zoom ou « dézoom » peut modifier complètement la représentation des objets affichés sur l'écran. Il serait donc malaisé de contrôler une ZUI avec des interacteurs intégrés dans la vue car ces interacteurs changeraient très souvent de position. Les utilisateurs ont donc besoin de moyens simples et uniformes pour interagir avec ces interfaces complexes. Nous proposons un nouveau type de menu, appelé « Control menu », qui permet un accès rapide aux nombreuses opérations qui sont nécessaires pour contrôler une ZUI efficacement. Ce menu peut intégrer l'équivalent de deux barres de défilement, ce qui permet aux utilisateurs de n'utiliser qu'un seul interacteur pour réaliser des opérations complexes en un seul geste. De fait les ZUIs présentées dans les sections suivantes n'utiliseront que cet unique interacteur.

Cet article présente d'abord nos nouvelles aides de navigation pour les ZUIs : les couches de contexte et d'historique, ainsi que la vue hiérarchique. Nous introduisons ensuite les Control menus, avant de terminer avec une description de « Zomit », l'outil de développement d'interfaces zoomables que nous avons réalisé pour tester les idées présentées dans cet article.

Nous avons conservé la terminologie anglo-saxonne d'origine dans les cas où une traduction française spécifique ne s'est pas encore imposée afin d'éviter d'employer des expressions peu satisfaisantes ou peu compréhensibles.

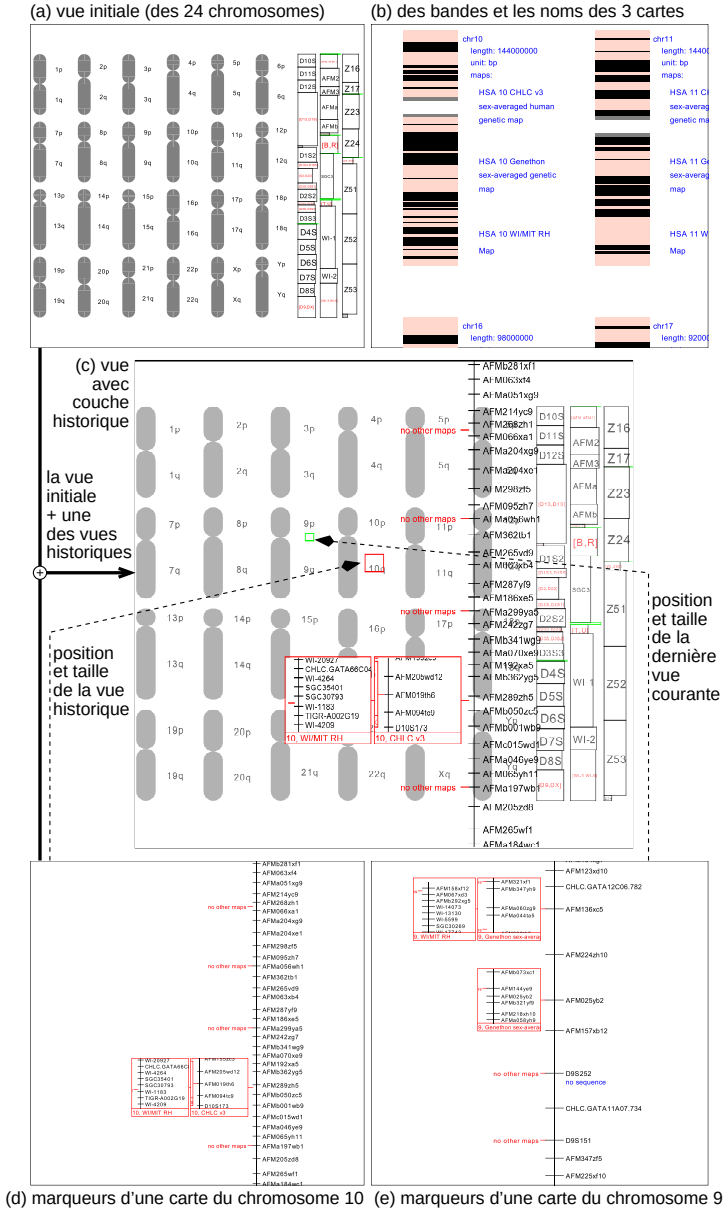

Figure 1 : quelques vues de ZoomMap et de sa couche historique

2 Couche de contexte

Avec une ZUI, un utilisateur ne voit qu'une vue à la fois : le focus. Le contexte a été perdu. D'autres types d'interfaces telles que les vues « fisheye » [Furnas 1986] et le « Document Lens » [Robertson et Mackinlay 1993] intègrent le focus et le contexte en affichant une partie de l'information qui entoure le focus. Ces techniques déforment la représentation graphique de l'espace d'information en éliminant certains objets ou en modifiant leur taille ou leur position. D'autres techniques proposent une vue du contexte affichée à côté de la vue du focus (un multiplexage spatial) ou bien une vue du contexte affichée à la place de la vue du focus (un multiplexage temporel).

2.1 Notre couche de contexte

Contrairement à ces techniques, nous proposons une couche de contexte qui combine le focus et son contexte dans une seule fenêtre et sans déformation. Ceci peut être vu comme un multiplexage de la profondeur [Cox et al. 1998]. La couche de contexte est temporaire et affichée uniquement quand l'utilisateur le désire. Pendant son utilisation elle se superpose en transparence à la vue principale (le focus). L'affichage de cette couche est temporaire afin de ne pas surcharger l'écran quand l'utilisateur n'a pas besoin de voir le contexte. Elle disparaît dès que l'utilisateur termine le geste qui a provoqué son apparition à l'écran.

La couche de contexte peut être contrôlée dans deux directions : l'échelle du contexte (c'est-à-dire le niveau de zoom sémantique) et le niveau relatif de transparence des deux vues. Le réglage de l'échelle permet de montrer une vue contextuelle dont l'échelle peut varier de manière continuelle entre celle de la vue initiale et celle du focus courant. Ceci permet d'obtenir une « quantité de contexte » adéquate pour une vue focale donnée.

La figure 1d montre une vue que l'utilisateur peut voir après avoir navigué pendant quelques temps dans la ZUI. Cette vue ne contient pas d'élément permettant à l'utilisateur de savoir sur quelle carte de quel chromosome il se trouve. L'utilisateur peut alors afficher la couche de contexte, ce qui donne la figure 2a. Celle-ci est une superposition du contexte (figure 1a) sur le focus (figure 1d). La position du focus relative au contexte est indiquée par un rectangle situé au centre. Dans la figure 2a ce rectangle est sous le mot « 10q » et l'utilisateur sait donc que le focus montre une partie du chromosome 10. Dans la figure 2b, l'utilisateur a zoomé la couche de contexte (mais pas le focus) ce qui donne maintenant comme contexte une vue où les noms des cartes génétiques sont visibles. L'utilisateur peut voir le rectangle (indiqué par une flèche) qui rappelle la position du focus sur le texte « Généthon ». L'utilisateur sait donc que le focus montre actuellement cette carte génétique. L'utilisateur a pu voir le contexte à deux échelles différentes de manière à pouvoir situer le focus dans deux contextes différents.

Notre système permet d'autre part à l'utilisateur de se concentrer sur l'une des deux vues en changeant la transparence relative de ces deux vues. La transparence peut varier continuement d'un état où seul le focus est visible jusqu'à l'état opposé où seule la couche de contexte est visible. Par exemple, la figure 2c est similaire à la figure 2b sauf que l'utilisateur est maintenant en train de se concentrer sur le contexte et a légèrement fait disparaître le focus. Le rectangle qui rappelle la position du focus est toujours visible et l'utilisateur peut voir plus clairement que le focus montre actuellement la carte Généthon.

2.2 La différenciation entre les deux vues

Harrison et al. [1995b] ont testé l'utilisation de palettes transparentes d'outils sous forme d'icones superposées sur une fenêtre d'information. Cette surimposition crée deux couches : la couche avant contenant la palette d'icones et celle contenant la fenêtre d'information. La couche avant, la palette,

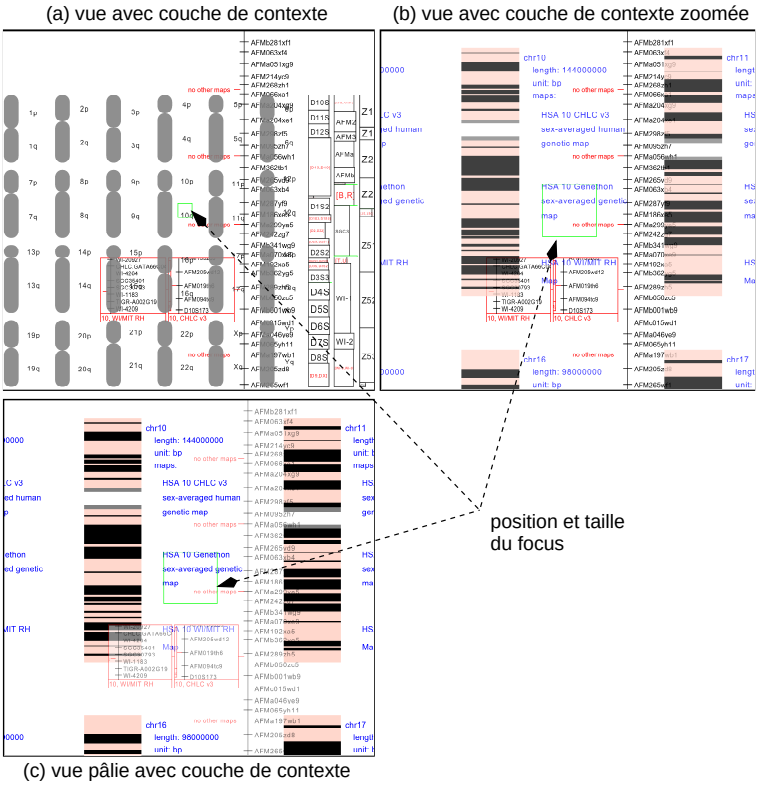

Figure 2 : la construction de la couche de contexte

est transparente afin de ne pas (complètement) cacher les objets situés sous la palette. Harrison et al. ont fait des expériences afin de trouver le niveau de transparence optimal. Leurs expériences présentaient une icone aux sujets pendant quelques secondes, puis, une palette de douze icones. La première icone était positionnée aléatoirement dans la palette et les sujets devaient la retrouver. Différents niveaux de transparence de la palette ont été testés afin de trouver le niveau le mieux adapté à la lecture des icones. Cette étude a montré qu'il est possible de lire et d'utiliser de telles palettes si le niveau de transparence est adapté à la tâche de l'utilisateur. Par ailleurs, Harrison et al. [1995a] ont proposé plusieurs façons de différencier deux vues surimposées sur l'écran : les vues peuvent contenir des couleurs différentes, des polices différentes, des contenus différents (par exemple graphiques ou textuels), etc.

Notre couche de contexte est différenciée de la vue du focus par le mouvement des objets sur l'écran et par leurs niveaux de transparence respectifs. Quand l'utilisateur change l'échelle du contexte les objets graphiques de cette couche bougent sur l'écran alors que ceux de la vue du focus ne bougent pas. Ce mouvement aide l'utilisateur à situer des objets, soit dans le contexte, soit dans la vue du focus. De plus, le changement interactif des niveaux de transparence n'affecte qu'une des vues à la fois, ce qui favorise également la différenciation des deux vues.

Cox et al. [1998] ont évalué un système où les utilisateurs doivent rechercher et connecter des tubes. Ce système affiche une vue détaillée, et donc partielle, de l'espace, sur laquelle est superposée une vue transparente de tout l'espace. Les utilisateurs peuvent se servir de ces deux vues pour trouver et déplacer les tubes. La vue globale est permanente ; elle est toujours affichée, même quand l'utilisateur n'en a pas besoin (ce qui peut éventuellement gêner l'utilisateur). Cette vue a un niveau

de transparence fixe. L'échelle de la vue globale est également fixe ce qui empêche l'utilisation de ce système dans les environnements où il n'existe pas de vue globale (tel que le Web) ou si la différence d'échelle entre la vue d'ensemble et la vue du détail est trop importante. Les auteurs ont testé l'utilisabilité de ce système avec différents niveaux de transparence de la vue globale. Ils ont mis en évidence que les utilisateurs trouvaient la vue globale utile pour des niveaux de transparence situés entre 50% et 70%.

Notre couche de contexte ressemble à la vue globale utilisé par ces auteurs dans la mesure où les deux systèmes surimposent une vue globale à une vue locale en utilisant un effet de transparence. Il est donc raisonnable de s'attendre, dans notre cas, à des performances d'utilisabilité comparables. Notre système comporte toutefois trois améliorations importantes. D'une part, notre couche de contexte existe temporairement lorsque l'utilisateur en a besoin. Son espace de représentation n'est donc encombré que de façon temporaire. D'autre part, le niveau de transparence de la couche de contexte peut être modifié interactivement par l'utilisateur : il peut donc choisir un niveau optimal en fonction de sa tâche courante. Enfin, comme expliqué précédemment, l'échelle de notre couche de contexte est également variable et le « mouvement » qui en résulte facilite la différenciation des deux couches.

Ce dernier point illustre l'importance des techniques d'interaction pour améliorer la compréhension des représentations visuelles. De même que pour le contrôle de l'échelle ou du niveau de transparence, il est vraisemblable que le « bouclage interactif » entre les actions de l'utilisateur et la réalisation d'effets visuels immédiats facilite la perception de certaines structures. Ceci est dû au fait que le système visuel humain est généralement plus efficace pour détecter des objets en mouvement. Cet effet devrait de plus être logiquement renforcé lorsque ce mouvement est lui-même contrôlé par l'utilisateur.

3 Couche historique

La couche de contexte décrite ci-dessus permet à l'utilisateur de trouver une réponse à la question « où suis-je ? ». Une autre question importante est « comment suis-je arrivé ici ? ». Les ZUIs nécessitent un mécanisme d'historique pour que l'utilisateur puisse retourner aux régions déjà visitées dans l'espace d'information afin de voir ces régions en relation avec le focus et la vue initiale. Nous proposons une autre vue temporaire appelée *couche historique*. Cette vue permet de se déplacer interactivement entre la vue initiale et la vue courante en suivant le chemin emprunté par l'utilisateur. Comme la couche de contexte, la couche historique est temporaire afin de ne pas surcharger l'écran.

De même que la couche de contexte, la couche historique est affichée en superposition de la vue courante du focus. Cette couche historique est contrôlée interactivement par l'utilisateur de telle sorte que le mouvement de la souris fasse apparaître successivement toutes les vues intermédiaires que l'on a préalablement affichées pour arriver à la vue courante. La *couche* historique (figure 1c) contient une des *vues* historiques (figure 1d) et se superpose à la vue initiale (figure 1a) ; cette combinaison s'affiche temporairement en lieu et place de la vue courante. La vue courante (figure 1e) n'est donc pas visible pendant l'utilisation de la couche historique. La position et la taille de la vue historique et de la vue courante sont indiquées sur la vue initiale par des rectangles de couleurs différentes (figure 1c). L'utilisateur, en se déplaçant dans la succession des vues historiques (figures 1a, b, d et e) peut donc revoir son parcours dans l'espace d'information et à tout moment mettre en relation la dernière vue courante avec une vue intermédiaire de son parcours.

4 Hiérarchie et navigation rapide

Les techniques présentées dans la section précédente facilitent la contextualisation du focus dans l'espace d'information. Cependant, elles n'informent pas l'utilisateur sur ce qui se trouve dans

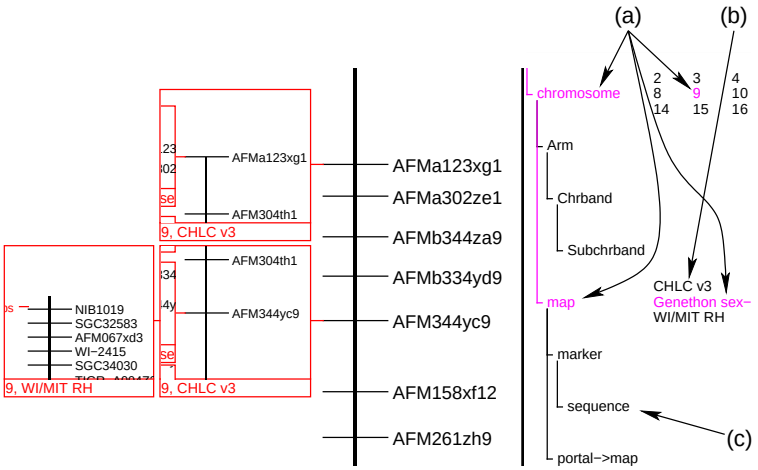

Figure 3 : notre interface zoomable avec la hiérarchie sur la droite

d'autres régions ni sur ce qui peut être trouvé en « zoomant » davantage. Les ZUIs sont souvent utilisées pour visualiser des données organisées hiérarchiquement mais l'utilisateur ne peut généralement pas utiliser cette organisation pour naviguer dans l'espace d'information. Par exemple, il est habituellement impossible de dézoomer automatiquement pour voir un objet en entier et d'utiliser la hiérarchie pour se déplacer d'un sous-objet vers un autre.

4.1 Une nouvelle vue hiérarchique

Les ZUIs fournissent une vue qui peut être considérée comme une tranche horizontale de l'espace 3D d'information (en considérant que la dimension verticale est celle du zoom). Nous proposons une seconde vue orthogonale à la première, qui est une tranche verticale « aplatie » de l'espace d'information. Cette seconde vue (figure 3) affiche les noms des objets qui sont situés au dessus de la vue courante dans la hiérarchie. Comme les objets ont également des types (un objet peut être un chromosome, une carte, une séquence, etc) la hiérarchie entière des types d'objets peut être montrée dans la seconde vue si l'espace d'information est suffisamment régulier. Sinon, seule une partie de la hiérarchie des types, centrée sur la position de l'utilisateur, peut être montrée. Cette seconde vue indique à l'utilisateur la structure de l'espace d'information, sa position courante, les autres informations disponibles, leur disposition spatiale, et comment les trouver.

Dans l'espace régulier de notre ZUI biogénétique, les chromosomes possèdent deux « bras », des données et des cartes. Cette structure est affichée dans la vue de la hiérarchie (figure 3) dès que la ZUI est lancée. La position de l'utilisateur dans la structure est indiquée en gris (en magenta sur l'écran) ; dans cet exemple l'utilisateur est en train de regarder la carte « Généthon » sur le chromosome 9 (figure 3a). Cette vue indique également que si l'utilisateur continue de zoomer sur la carte il trouvera les séquences des marqueurs génétiques (figure 3c).

La hiérarchie offre également à l'utilisateur un moyen efficace de se déplacer des objets visibles vers les objets non encore affichés qui leur sont liés. Par exemple, si l'utilisateur clique sur le mot « chromosome » dans la figure 3, la ZUI dézoomera suffisamment pour montrer le chromosome 9 en entier. L'utilisateur pourrait également cliquer sur les mots « CHLC v3 » (figure 3b), le nom de la carte à coté de la carte « Généthon » sur la chromosome 9, afin de se déplacer vers la même région sur la carte « CHLC v3 ».

4.2 Évaluation

Afin d'évaluer l'efficacité de la hiérarchie proposée dans cet article nous avons créé une version modifiée de notre ZUI sans la hiérarchie. Huit sujets, des volontaires parmi nos collègues d'Infobiogen, ont été formés sur notre ZUI. Ces volontaires étaient des informaticiens, des biologistes, et des assistants administratifs. Pendant l'évaluation il leur a été demandé de répondre à vingt-deux questions à choix multiples. La séance d'entraînement leur a montré comment répondre à ce genre de question avec et sans la vue de la hiérarchie. Les sujets ont été divisés en quatre groupes et ont traité onze questions avec aide de la hiérarchie, et onze sans. Deux groupes ont commencé par répondre à onze questions avec l'aide de la hiérarchie, les deux autres groupes ont commencé sans cette aide. Ainsi chaque question a été traitée avec l'aide de la hiérarchie par deux groupes de sujets, et sans par les deux autres groupes. De plus, dans chaque cas, les deux séries de onze questions ont été présentées dans un ordre différent à chacun des deux groupes afin d'éviter que l'ordre des questions n'influe sur les résultats.

Les questions étaient du type « quel est le nom du dernier marqueur de la carte Généthon des chromosomes 1, 8, 13 ? ». Celles-ci ne concernaient pas directement la structure de l'espace d'information mais une connaissance de cette structure (fournie par l'entraînement) aidait les sujets à répondre plus rapidement aux questions.

Pour chaque sujet nous avons calculé le temps pris pour répondre aux onze questions sans la vue hiérarchique divisé par le temps pris pour répondre aux onze questions avec la vue hiérarchique. Une valeur supérieure à 1 signifiait donc que la vue hiérarchique aidait à l'exécution des tâches demandées. La moyenne était de 1,58 avec un écart type de 0,54. Ce grand écart type venait du fait que certains utilisateurs ne connaissaient pas la structure de l'espace d'information avant cette étude. Ces utilisateurs trouvaient que la séance d'entraînement n'était pas suffisante et donc que la seconde partie des questions était plus facile. Cependant, les sujets étaient en général plus rapides avec la vue hiérarchique et les commentaires étaient positifs.

4.3 Autres techniques

La technique d'« Excentric Labeling » [Fekete et Plaisant 1999] permet d'identifier des objets sur l'écran. Cette technique étiquette, au moyen de bulles d'aide situées dans la vue principale, les objets se trouvant autour du pointeur. Nous proposons une autre technique, non intrusive, pour identifier les objets autour du pointeur. Quand l'utilisateur déplace le pointeur sur la vue principale, la vue de la hiérarchie est mise à jour en affichant le type et le nom de l'objet qui est situé sous le pointeur. Si le pointeur quitte la vue principale ou s'il n'est pas sur un objet, la vue hiérarchique indique le plus bas niveau dans la hiérarchie auquel tous les objets visibles appartiennent. Cette technique d'étiquetage présente des similarités avec les bulles d'aide dans la mesure où l'utilisateur n'a pas besoin de demander l'information explicitement. Elle a toutefois pour avantage de ne pas empiéter sur la vue principale.

L'interface du système gIBIS [Conklin et Begeman 1988] inclut une vue globale du graphe IBIS. Cette vue montre tous les nœuds du graphe (organisés selon leur liens primaires) ainsi que l'un de leurs attributs (le sujet du nœud). Elle est généralement trop grande pour être affichée en entier ce qui nécessite d'utiliser des ascenseurs pour naviguer. Quand l'utilisateur se déplace dans le graphe, en zoomant ou en défilant, la vue globale se déplace afin de montrer la position de l'utilisateur dans le graphe. Cette vue peut également être utilisée pour se déplacer dans le graphe car l'utilisateur peut cliquer sur un nœud sur lequel il souhaite recentrer le focus. Notre hiérarchie se distingue de cette vue globale du fait qu'elle ne montre que la structure de l'espace (au lieu de l'espace en entier) et les noms des objets situés hiérarchiquement « au-dessus » des objets visibles. Ainsi, même les ZUIs de grande taille (mais structurées) peuvent tenir dans un espace réduit et des ascenseurs ne sont pas nécessaires.

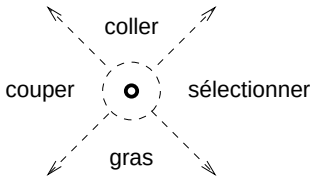

Figure 4 : un « Pie menu » et, en pointillé, ses zones actives

Comme une ZUI contient typiquement un très grand nombre d'objets, une vue exhaustive de tous les objets serait trop grande pour être utilisable.

5 Un nouveau type de menu : le « Control menu »

Cette section présente un nouveau type de menu contextuel, nommé « Control menu ». Ce nouvel interacteur est particulièrement bien adapté pour contrôler des interfaces complexes comme les ZUIs. Son usage reste cependant tout à fait général et n'est pas limité à ce type d'interface (comme nous le verrons dans la section suivante).

5.1 Interacteurs habituels

Des menus standards tels que les menus déroulants et les menus contextuels permettent aux utilisateurs de choisir facilement des opérations à effectuer. Les menus contextuels sont activés à un endroit choisi par l'utilisateur dans l'interface. L'interface peut alors adapter le contenu du menu à la position choisie et associer l'action choisie à l'objet se trouvant à cette même position. Par ailleurs, de nouveaux menus contextuels, les « Pie menus » [Callahan et al. 1988 ; Hopkins 1991] et les « Marking menus » [Kurtenbach et Buxton 1994], ont été proposés afin de rendre l'utilisation des menus contextuels plus rapide.

La différence la plus importante entre un Pie menu et un menu contextuel standard est le fait que les entrées dans un Pie menu sont distribuées autour du centre du menu (figure 4). L'utilisateur n'a plus besoin de sélectionner une ligne dans une liste linéaire mais peut tout simplement déplacer le pointeur dans la bonne direction et relâcher le bouton de la souris. Les Pie menus sont par exemple utilisés dans le jeu « The Sims » (http://www.thesims.com) pour contrôler rapidement les actions des personnages. Ces menus contextuels permettent d'associer des actions aux personnages sans sélectionner préalablement le personnage. Ainsi, un joueur expert peut faire des actions rapidement car il n'est pas obligé d'attendre que le menu s'affiche pour y sélectionner une action. Un Marking menu ressemble à un Pie menu sauf que chaque fois que l'utilisateur active une opération, le système dessine un trait sur l'écran indiquant le geste optimal que l'utilisateur aurait pu faire.

La loi de Fitts [MacKenzie 1995 ; Raskin 2000] donne le temps nécessaire pour déplacer le curseur vers un bouton sur l'écran. Ce temps est fonction de la taille du bouton et la distance entre le curseur et ce bouton selon la formule $t = a + b \log_2(d/s + 1)$ où s est la taille du bouton et d la distance entre le curseur et le bouton. Cette loi dit donc que les menus contextuels devraient être rapides à utiliser car l'utilisateur n'est pas obligé de se déplacer pour se servir du menu. D'après cette même loi, les Pie et Marking menus sont encore plus avantageux car, pour choisir une opération, il suffit de déplacer le curseur vers une des zones actives du menu (indiquées en pointillé dans la figure 4). Ces zones sont grandes et proches du curseur (le menu s'affiche de sorte que le centre du menu soit sous le pointeur). Rapidement l'utilisateur apprend la position des opérations les plus fréquemment utilisées et peut alors se servir du menu sans avoir à le regarder. Cette facilité de sélection contraste

avec le temps pris pour sélectionner la bonne ligne dans un menu contextuel de type linéaire.

5.2 Opérations avec des paramètres continus

Une opération continue est une opération comportant au moins un paramètre de type numérique (avec un nombre important de valeurs possibles). La sélection de l'échelle d'une ZUI est un exemple d'opération continue. Les menus contextuels (les Pie et Marking menus ainsi que les menus contextuels standards) ne permettent pas aux utilisateurs de contrôler l'opération choisie de façon continue (par exemple pour effectuer un défilement ou zoomer jusqu'à ce que l'utilisateur trouve la bonne taille). Ils ne permettent pas non plus aux utilisateurs de fournir des paramètres pour contrôler l'opération sélectionnée. Par exemple, une opération telle que le changement de la taille de police dans un traitement de texte nécessite souvent l'ouverture d'une boîte de dialogue pour entrer un paramètre : la nouvelle taille. Les utilisateurs doivent d'abord ouvrir le menu et sélectionner la bonne opération puis se concentrer ensuite sur un deuxième interacteur, typiquement une boîte de dialogue. Une fois la nouvelle taille entrée, la boîte de dialogue disparaît et l'utilisateur doit à nouveau changer de contexte. Cette interaction nécessite plusieurs changement de focus et des déplacements de souris qui ralentissent l'interaction.

Ce problème existe par exemple dans des applications telles qu'Acrobat Reader d'Adobe pour contrôler des opérations continues telles que zoomer et défiler. Cette application propose trois façons différentes de zoomer : via une boîte de dialogue, via un « option menu », ou en cliquant avec la souris mais après avoir sélectionné un mode *ad hoc*. L'utilisation d'un « option menu » ou d'une boîte de dialogue provoque les problèmes d'interaction précédemment cités. De plus l'« option menu » occupe une place non négligeable sur l'écran. Ces deux interacteurs ne sont pas contextuels et ne permettent donc pas à l'utilisateur d'indiquer sur quelle région de l'écran il veut centrer le zoom. Il sera donc nécessaire de recentrer la vue affichée après chaque zoom. Dans le troisième cas, l'utilisation de la souris pour zoomer minimise les changements de point d'attention et évite d'avoir à recentrer la vue mais nécessite par contre d'avoir préalablement sélectionné un mode spécifique (car le bouton de la souris sert également à d'autres types d'actions dans d'autres modes). De plus, ces trois interacteurs changent l'échelle par incréments prédéfinis et l'utilisateur devra donc les activer plusieurs fois avant d'obtenir l'échelle désirée. Les mêmes problèmes se posent pour faire défiler le document à l'aide de barres de défilement qui utilisent encore plus de place que l'« option menu » (elles fournissent néanmoins une information sur la position de la vue courante dans le document).

L'opération de déplacement dans une ZUI nécessiterait l'usage d'un bouton spécifique de la souris (si elle en a plusieurs) ou l'usage conjoint de ce bouton et d'une touche de contrôle du clavier ou encore d'utiliser deux ascenseurs. On ne peut se déplacer avec un menu standard sauf en répétant des commandes peu ergonomiques telles que « déplacer un peu vers la gauche ». Et il faut de plus pouvoir aisément contrôler le niveau de zoom.

5.3 « Control menu »

Nous proposons un nouveau type de menu contextuel, appelé « Control menu » [Pook et al. 2000], qui permet à l'utilisateur de fournir jusqu'à deux paramètres à l'opération choisie ou de la contrôler dans une ou deux dimensions indépendantes. Ce menu permet ainsi aux utilisateurs de *choisir* et de *contrôler* des opérations en un seul geste.

Un Control menu a des similarités de comportement avec un Pie menu. Un utilisateur, qui ne sait pas où se trouve l'opération qu'il désire, enfonce le bouton de la souris, attend 0,3 secondes (temps expérimental proposé par [Kurtenbach et Buxton 1994]) jusqu'à ce que le menu soit affiché centré sous le curseur, puis déplace ce dernier dans la direction de l'opération désirée (figure 5). Le menu disparaît et l'opération commence dès que le curseur a été déplacé de la *distance d'activation*

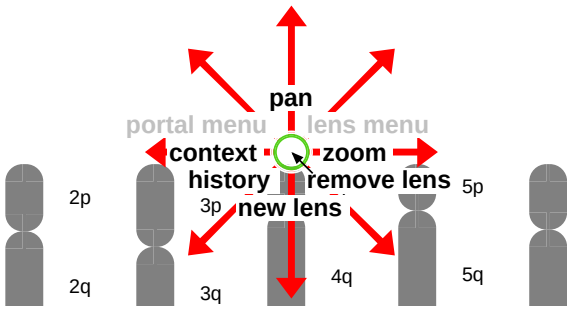

Figure 5 : un Control menu (sur la vue de la figure 1a)

depuis le centre du menu (nous avons empiriquement choisi une distance d'activation de cinq fois le rayon du cercle au centre du menu). L'opération se termine quand l'utilisateur relâche le bouton de la souris. Les mouvements de la souris effectués pendant l'opération fournissent les paramètres nécessaires au contrôle de cette opération. Un utilisateur qui sait où se trouve la commande souhaitée fait le même geste qu'un novice mais sans effectuer la pause qui fait apparaître le menu. Ainsi, les utilisateurs experts ne sont pas distraits par l'apparition du menu et les utilisateurs novices apprennent progressivement le geste expert.

Un Control menu comprend jusqu'à huit flèches et un libellé par flèche. Les libellés sont désignés en caractères noir sur fond blanc (ou du caractères gris pour les choix non-appliquables à ce moment). Outre ces éléments, le menu est transparent afin de ne pas cacher la zone d'intérêt lorsque l'utilisateur est en train de choisir l'opération.

5.4 Opérations contrôlées par un seul paramètre

Une entrée dans un Control menu peut par exemple servir à modifier le niveau de zoom dans une ZUI. Cette opération illustre l'intégration d'un menu et d'une barre de défilement dans un seul interacteur. La figure 6 montre les mouvements de la souris pendant l'utilisation d'un Control menu pour choisir et contrôler une opération de zoom ou de dézoom. L'utilisateur enfonce le bouton de la souris et déplace celle-ci de la distance d'activation (le mouvement numéro 1 dans la figure 6) vers la droite (l'opération « zoom » étant sur la droite du Control menu représenté à la figure 5). Ceci amorce l'opération de zoom et la forme du pointeur change à ce moment sur l'écran. À partir de ce moment, les mouvements de la souris vers la droite (mouvements 2 et 4 dans la figure 6) zooment la vue et les mouvements vers la gauche (mouvement 3) la dézooment. Le contrôle par retour d'information est immédiat : la vue change dès que l'utilisateur bouge la souris. L'utilisateur relâche le bouton de la souris lorsque l'échelle voulue (un dézoom dans la figure 6) a été obtenue. Cet exemple montre qu'avec le menu de la figure 5, pour dézoomer il faut d'abord zoomer un peu. Alternativement, une

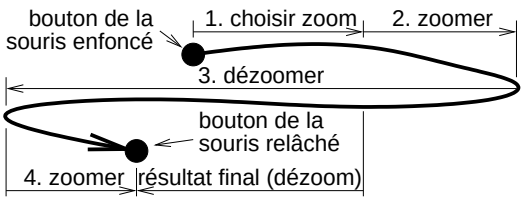

Figure 6 : zoomer avec un Control menu

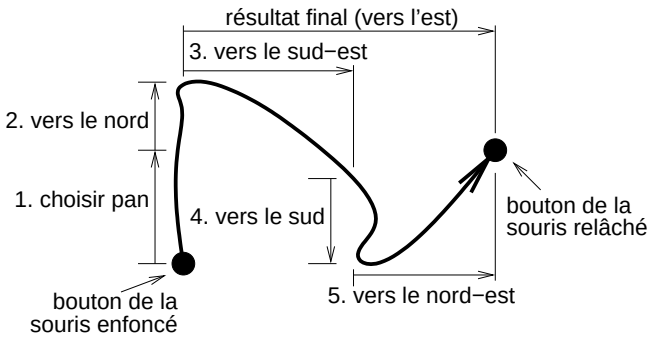

Figure 7 : défilement avec un Control menu

entrée « dézoomer » pourrait être ajoutée dans la partie gauche du menu. L'utilisateur déplacerait alors directement la souris vers la gauche pour dézoomer. Cette solution a cependant le désavantage d'utiliser davantage de place dans le menu et sa mise en œuvre est donc dépendante des spécificités de l'application.

Pendant l'opération de zoom l'utilisateur peut annuler cette opération en déplaçant, sur une grande distance, la souris dans la direction orthogonale (vers le haut ou vers le bas dans le cas de l'exemple). L'utilisateur peut alors confirmer l'annulation en relâchant le bouton de la souris ou infirmer cette annulation en retournant la souris vers sa position verticale initiale. Dans ce cas, le bouton de la souris reste enfoncé et l'utilisateur peut continuer de zoomer. Il n'est pas possible d'annuler toutes les opérations de cette manière ; seules les opérations où une seule des deux directions de déplacement de la souris est utilisée peuvent être annulées de cette manière.

5.5 Opérations contrôlées par deux paramètres

Un Control menu peut aussi être utilisé pour effectuer des défilements bi-directionnels. Il remplace alors deux barres de défilement. L'opération de défilement est sélectionnée en enfonçant le bouton de la souris et en déplaçant la souris vers le haut (« pan » sur la figure 5 en haut du menu). La vue suit le curseur pendant l'opération (figure 7). Cette opération ne peut pas être annulée pendant l'opération car les deux sens de déplacement de la souris ont déjà une signification.

Il est possible de faire la même remarque que sur l'opération « dézoomer » ; si l'utilisateur veut déplacer la vue vers le bas, il faut d'abord qu'il la déplace légèrement vers le haut. Nos observations informelles indiquent que ceci semble poser moins de problèmes que la nécessité de zoomer un peu afin de pouvoir dézoomer. Ceci est probablement dû au fait que le changement d'échelle du zoom est généralement vu comme une opération relative (typiquement contrôlée par le déplacement d'un curseur sur un potentiomètre autour d'une origine) tandis que le positionnement sur une surface plane est une opération d'une autre nature (et qui s'applique directement sur la surface affichée via la notion de pointage).

Un Control menu est bien adapté pour contrôler deux paramètres intégraux. Deux paramètres sont intégraux [Jacob et Sibert 1992] lorsque leurs attributs se combinent dans la pensée de l'utilisateur en un seul attribut composé. Par exemple, les coordonnées x et y d'un objet sont intégrales car les utilisateurs les combinent et les considèrent comme la position de l'objet. Un déplacement diagonal de la souris a alors une signification simple : déplacer l'objet sur la diagonale. Par contre, la taille et la couleur d'un objet ne sont pas deux paramètres intégraux. Si un Control menu était utilisé pour contrôler de tels paramètres simultanément, un déplacement diagonal de la souris n'aurait pas de signification immédiate.

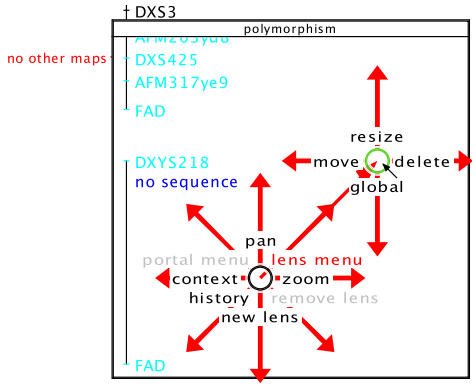

Figure 8 : le sous-menu des lentilles dans notre interface zoomable

5.6 Boutons et sous-menus

Les Control menus peuvent également contenir des commandes simples qui n'ont pas de paramètre. Dans ce cas les commandes qui peuvent être annulées sont exécutées dès que le curseur a été déplacé de la distance d'activation depuis l'endroit où le bouton de la souris a été enfoncé. Puisque l'utilisateur maintient le bouton de la souris enfoncé, un mouvement du curseur dans l'autre direction annule l'opération. L'utilisateur peut alors infirmer l'annulation en déplaçant à nouveau le curseur dans l'autre sens. L'opération ne devient définitive que lorsque le bouton de la souris est relâché (comme avec les « boutons poussoirs » habituels).

Un Control menu peut avoir des sous-menus. L'utilisateur enfonce le bouton de la souris et déplace le curseur dans la direction de l'entrée dans le Control menu correspondant au sous-menu désiré. La figure 8 montre le sous-menu qui permet à l'utilisateur de contrôler des lentilles affichées dans la ZUI. Ce sous-menu peut être affiché lorsque le pointeur est sur une lentille (la lentille « polymorphism » dans la figure 8). Le sous-menu est visuellement attaché au menu principal par une ligne rouge avec deux flèches et l'utilisateur choisit une opération dans le sous-menu en déplaçant le pointeur dans la direction de son entrée dans le sous-menu.

5.7 Control menus versus Marking menus

Avec un Marking menu la distance parcourue par le curseur n'a pas d'importance. Seule la forme du mouvement est signifiante et celle-ci est analysée une fois que le bouton de la souris est relâché (ou quand l'utilisateur arrête de bouger le curseur afin d'ouvrir un sous-menu). Avec un Control menu la distance parcourue par le curseur est informante. La position du curseur est constamment analysée et l'opération commence dès que la distance d'activation a été atteinte.

5.8 Analyse des propriétés des Control menus

Le modèle d'interaction de Beaudouin-Lafon [2000] définit un espace d'analyse qui peut être utilisé pour comparer de nouveaux interacteurs aux interacteurs déjà connus.

Un Control menu est contextuel : il agit soit sur l'objet, soit sur la position se trouvant sous le curseur. Il ne nécessite pas non plus de se déplacer vers une barre de menu. L'*écart spatial* (la distance sur l'écran entre le menu et l'objet sur lequel il agit) est donc nul. Un Control menu ressemble aux poignées des objets dans un logiciel de dessin graphique ou à l'action de glisser/coller (« drag and drop »). Lors du zoom et du défilement, l'écart temporel (le temps entre le mouvement de

la souris et la réaction de l'objet manipulé) est également nul car l'interface réagit immédiatement quand l'utilisateur déplace la souris. De ce point de vue, un Control menu ressemble à une poignée ou à une barre de défilement (si elle réagit immédiatement aux mouvements de la souris). Le degré d'indirection (qui combine ces deux écarts) est donc faible.

Le ratio entre le nombre de degrés de liberté fourni par notre interacteur et le nombre de degrés de liberté de la souris (le degré d'intégration) est de 2/2 car, pour toutes les interactions dans notre ZUI, les deux degrés de liberté de la souris sont utilisés. Pour faire défiler la vue dans un logiciel avec des barres de défilement, il est nécessaire d'en utiliser deux car chaque barre n'utilise qu'un des deux degrés de liberté de la souris. Par contre, un Control menu utilise les deux degrés de liberté de la souris et un seul Control menu est donc suffisant pour faire défiler la vue dans toutes les directions.

Le degré de compatibilité (la similarité entre l'action physique de l'utilisateur et la réponse de l'objet sur lequel il agit) est élevé pendant un défilement car les mouvements de la souris sont directement reflétés par les mouvements de la vue. L'utilisateur déplace l'objet sous le curseur au moment du début de l'opération jusqu'à sa nouvelle position. Cette action présente un même niveau de compatibilité que le glisser/coller ou que de tirer sur une poignée. Ce degré de compatibilité est moins élevé pendant un zoom du fait de la possibilité d'annulation. En effet, tandis que les mouvements horizontaux sont en relation directe avec les mouvements de la souris, l'annulation requiert des mouvements verticaux, ce qui peut paraître moins évident à l'utilisateur. Des utilisateurs ont également suggéré que le choix des mouvements horizontaux pour contrôler le niveau de zoom était non optimal et que des mouvements verticaux seraient plus naturels pour réaliser cette opération.

Cette analyse permet de constater que les Control menus partagent de nombreux attributs avec les poignées des objets graphiques.

Enfin, un Control menu a les mêmes avantages qu'un Pie menu d'après la loi de Fitts. L'utilisateur n'a pas besoin de bouger la souris pour faire afficher le menu, le menu s'affiche sous le pointeur, et les zones de l'écran sur lesquelles il faut déplacer le pointeur pour sélectionner une opération sont grandes et proches du pointeur.

6 Applications

Un Control menu est un nouveau type de menu contextuel qui peut être utile dans de nombreux types d'applications. Nous l'avons utilisé dans deux exemples de ZUIs et dans une interface de monde virtuel. Nous donnons aussi des indications sur la manière dont il pourrait être utilisé dans un traitement de texte.

6.1 Contrôle des couches de contexte et d'historique

Un Control menu est utilisé pour contrôler les couches de contexte et d'historique dans notre ZUI. Ainsi, la couche de contexte a-t-elle deux paramètres : l'échelle et le niveau de transparence. L'échelle est contrôlée par les mouvements horizontaux de la souris et la transparence par les mouvements verticaux. Le contrôle de la couche historique est similaire : la position sur le chemin de l'utilisateur est contrôlée par les mouvements horizontaux et la transparence par les mouvements verticaux. Ces paramètres ne sont pas intégraux car les deux paramètres pris ensemble n'ont pas un sens simple. Un mouvement diagonal de la souris n'a alors pas de sens intrinsèque. Nous examinons actuellement si cela crée des problèmes aux utilisateurs et si l'option qui consisterait à ne prendre en compte que la composante principale des mouvements diagonaux aiderait à résoudre cette difficulté éventuelle.

Comme expliqué précédemment, les couches de contexte et d'historique n'existent que pendant l'exécution du geste qui fait apparaître le Control menu ; dès que l'utilisateur relâche le bouton de la souris, la couche transparente disparaît.

Control menus

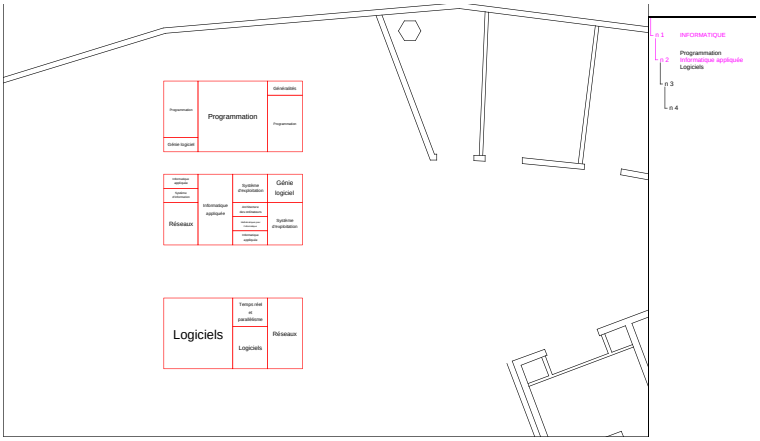

Figure 9 : vue globale de la bibliothèque

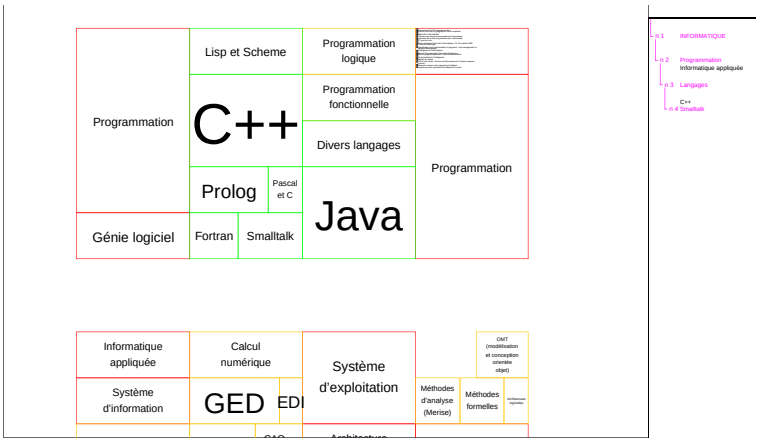

Figure 10 : une vue des rayons de la bibliothèque

6.2 Bibliothèque virtuelle

En plus de l'interface de la base HuGeMap, notre outil de développement d'interfaces zoomables, nommé Zomit, a été utilisé pour créer l'interface d'une bibliothèque virtuelle. Cette ZUI, qui permet de naviguer dans les rayons d'une bibliothèque, a été développée à l'École de Mines de Nantes [Lecolinet et al. 2001]. L'objectif de ce projet était de comparer la vitesse de la recherche de livres en utilisant une ZUI, une interface tridimensionnelle, et une bibliothèque réelle. Cette ZUI permet de zoomer d'une vue globale de la bibliothèque (figure 9) à une vue des différentes matières (figure 10) puis à une vue des livres contenus dans les différents « rayons » (figure 11). Cette bibliothèque contient plus de trois mille livres groupés dans des rayons, eux-mêmes regroupés par sujets. Cette interface contient également les images de couverture des livres, qui sont affichées quand l'utilisateur zoome dessus.

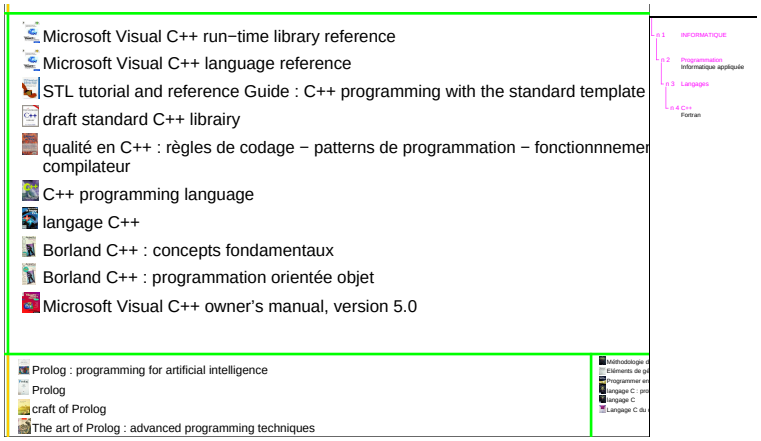

Figure 11 : une partie les livres de la bibliothèque

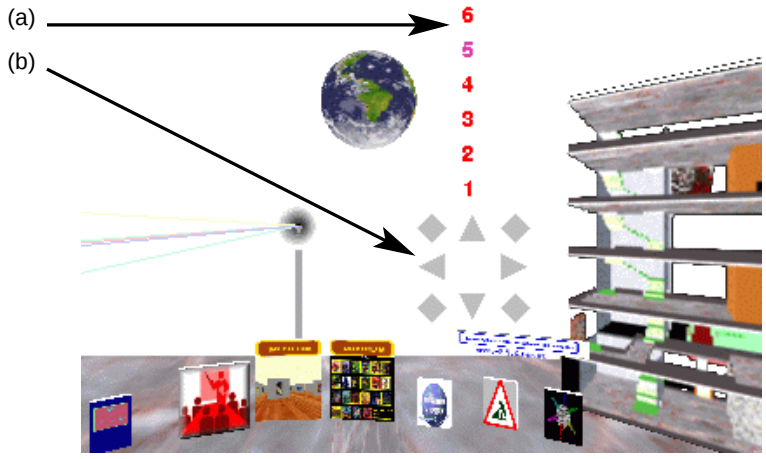

Figure 12 : le Control menu dans vreng ; (a) le guide, et (b) le Control menu

6.3 Interaction dans un monde virtuel

Un Control menu a également été utilisé pour se déplacer dans le monde virtuel *vreng* (figure 12) disponible à l'URL http://www.infres.enst.fr/net/vreng. Ce Control menu a une forme légèrement différente de celle utilisée dans les ZUIs car ici le Control menu est principalement utilisé pour se déplacer. Il n'est donc pas nécessaire d'afficher du texte dans le menu, les opérations disponibles (qui contrôlent la vitesse de déplacement dans diverses directions) étant indiquées par des flèches (figure 12b). À la différence du cas précédent un guide visuel est affiché une fois l'opération sélectionnée. Ce guide indique la liste des vitesses disponibles et, par conséquent, la distance nécessaire de déplacement de la souris pour choisir la vitesse souhaitée.

6.4 Traitements de texte et logiciels de présentations

Dans ces logiciels, la souris est typiquement utilisée pour sélectionner du texte. Un Control menu permet de choisir une opération parmi plusieurs opérations puis de contrôler celle-ci. On pourrait

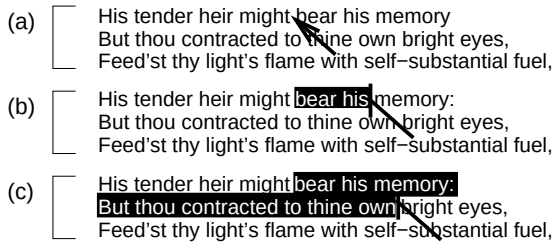

Figure 13 : sélectionner du texte avec un Control menu

ainsi par exemple sélectionner et modifier du texte en un seul geste.

Quand un Control menu est utilisé dans un de ces logiciels, l'opération « sélectionner » est placée à la droite du Control menu (c'est-à-dire qu'elle remplace l'opération « zoom » de la figure 5). Dans la figure 13a l'utilisateur a indiqué le début du texte devant être sélectionné en enfonçant le bouton de la souris lorsque le pointeur était entre les mots « might » et « bear ». Il déplace alors le pointeur de la distance d'activation vers la droite. À ce moment l'opération de sélection de texte commence et le texte situé entre la position initiale du pointeur et sa position actuelle est sélectionné (figure 13b). La sélection de texte se fait alors normalement jusqu'à ce que l'utilisateur relâche le bouton de la souris.

Les opérations nécessitant de fournir des paramètres (des tailles de police par exemple) sont décomposées en deux gestes. Le texte à modifier est sélectionné comme décrit ci-dessus puis le menu est de nouveau utilisé pour exécuter l'opération de changement de taille.

Des opération telle que « couper », « souligner », et « mettre en gras » peuvent être réalisées de la même manière (deux utilisations du menu) ou via une seule entrée dans le Control menu. Dans ce dernier cas, l'opération choisie sera effectuée au fur à mesure (cas de la mise en gras) ou à la fin (cas du « couper ») de la sélection de texte. Ceci donne un retour de contrôle encore plus direct et permet d'exécuter des commandes simples en minimisant l'usage des menus.

Un Control menu pourrait également être utilisé dans un logiciel de préparation de présentations. Ce cas est un peu différent du cas précédent dans la mesure ou la taille et forme du texte sont choisies plus librement dans une présentation afin que l'utilisateur puisse créer l'effet visuel souhaité. Dans le cas d'un traitement de texte, la taille et la forme sont souvent précisées par un style predéfini. Un guide visuel serait alors nécessaire pour choisir le style approprié, comme dans le cas du monde virtuel (section 6.3). Par contre, dans le cas d'un éditeur de présentation, l'utilisateur pourrait directement changer la taille des polices de manière relative comme dans le cas des interfaces zoomables. Le retour visuel est alors immédiat ce qui permet de choisir l'effet voulu en une seule interaction. Les autres paramètres de formatage (couleur du texte, mise en gras, indentation, etc.) pourraient également être sélectionnés de la même façon (via l'utilisation ou non d'un guide visuel selon le cas.)

7 Implémentation

Les techniques d'interaction présentées dans cet article ont été réalisées et testées dans une ZUI nommé ZoomMap [Pook et al. 1998], de type client/serveur, conçue pour être utilisée sur Internet. ZoomMap visualise les données de la base HuGeMap. Celle-ci est stockée dans une base de données objet gérée par le système EyeDB [Viara et al. 1999].

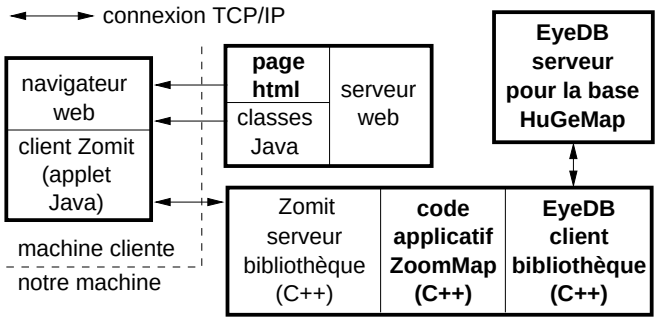

Figure 14 : implémentation client/serveur

7.1 L'outil de développement de ZUIs : Zomit

ZoomMap repose sur l'outil de développement Zomit. Zomit implémente toutes les fonctionnalités de base d'une ZUI, telles que le zoom sémantique [Furnas et Bederson 1995], les portails et les lentilles magiques. Un portail est positionné dans le monde virtuel par le développeur d'une ZUI et contient une vue d'une autre région du monde virtuel. L'utilisateur peut zoomer et faire défiler la vue dans le portail. Une liste de lentilles magiques possibles est proposée par la ZUI et l'utilisateur peut en créer une nouvelle quand il le souhaite et ensuite la déplacer. Une lentille magique contient une transformation, propre à la lentille, de la région du monde virtuel qu'elle couvre.

Zomit consiste en un client Java et une bibliothèque C++ qui communiquent par une connexion TCP/IP. La figure 14 montre la structure d'une ZUI construite avec l'outil de développement Zomit. Quand l'utilisateur lit une page Web qui fait référence à un client Zomit, l'*applet* Java est chargée dans le navigateur de l'utilisateur puis se connecte au serveur Zomit (qui doit se trouver sur la machine ayant envoyé la page). Le serveur Zomit lit la base de données et crée les objets du monde virtuel. Ce programme se charge de stocker ces objets et de les envoyer au client quand nécessaire.

Le client s'exécute dans un navigateur Web standard et ne nécessite pas d'installation préalable de la part de l'utilisateur. L'*applet* est complètement générique et le même code Java peut être utilisé pour communiquer avec n'importe quel serveur Zomit. L'*applet* peut aussi être installée en tant qu'application Java. Dans ce cas, le même client peut être utilisé pour communiquer avec n'importe quel serveur Zomit sur n'importe quelle machine.

Lorsque le client envoie au serveur la position de l'utilisateur dans l'espace d'information, le serveur répond en lui envoyant les objets graphiques qu'il doit afficher à cette position. Ces objets sont simples à dessiner et nécessitent peu de bande passante lors de leur transmission. Le client stocke les objets en mémoire locale et peut donc répondre rapidement aux changements d'échelle ou de position qui redemandent l'affichage de ces objets ; seul l'affichage des objets nouveaux est ralenti par la latence de la connexion avec le serveur. L'interaction avec des lentilles magiques et les portails est également gérée par le client ; seul l'affichage de nouveaux objets dans ces interacteurs peut éventuellement être ralenti.

La transparence des couches est simulée dans l'*applet* Java au moyen de fonctions XOR car les commandes graphiques utilisées pour réaliser la transparence ne sont pas encore disponibles dans de nombreux navigateurs. La fonction XOR est également nettement plus rapide que l'affichage de couches vraiment transparentes par composition de couches (ou « alpha blending »). La transparence est par contre effective quand le client tourne comme une *application* Java en utilisant les bibliothèques Java 2.

Le serveur s'exécute sur la même machine que la base de données afin de pouvoir lire rapidement les nombreuses informations requises pour construire les objets graphiques à envoyer au client. De

même que le client, la bibliothèque du serveur est générique ; il n'est donc pas nécessaire de la modifier pour visualiser un autre type de données. Seul le code qui lit la base de données et celui qui appelle la bibliothèque, en texte gras dans la figure 14, doivent être modifiés pour une application particulière.

7.2 Le développement d'un serveur Zomit

Le serveur d'une ZUI utilise la bibliothèque C++ Zomit et doit créer, au moyen de celle-ci, des instances d'une classe. Chaque instance est enregistrée comme couvrant une région en trois dimensions (x, y, et le niveau d'échelle) du monde virtuel. Quand le client demande des objets dans une partie de la région associée à une instance (car l'utilisateur y est entré), le code de cette instance est appelé. Ce code peut créer des objets graphiques, exclusivement situés dans la région initialement associée avec l'instance, et enregistrer d'autres instances de la classe couvrant les sous-régions de la région initiale. Ces sous-régions sont normalement celles que l'utilisateur verra en zoomant d'avantage. Cette technique, où les objets graphiques ne sont pas générés avant d'en avoir besoin, est une forme d'évaluation paresseuse. Un serveur peut ainsi être utilisé pour visualiser un monde virtuel de très grande taille où il n'est pas possible de générer (et stocker) tous les objets graphiques par avance.

Chaque objet a une position en coordonnées absolues dans le monde virtuel et deux niveaux d'échelle entre lesquelles il est déclaré visible. L'utilisateur voit une partie rectangulaire du monde à une échelle donnée. Quand il zoome ou dézoome ce niveau d'échelle change. Ainsi quand l'utilisateur zoome sur un objet il devient de plus en plus grand jusqu'au moment où il n'est plus visible. Le concepteur peut placer les objets arbitrairement mais fera généralement en sorte qu'un objet qui disparaît soit remplacé par d'autres objets représentant la même information mais de manière plus détaillée.

Un portail est un objet semblable à un rectangle mais avec les coordonnées de la partie de monde virtuel à afficher. L'affichage du contenu d'un portail, le défilement et le zoom dans le portail sont gérés par le client sans intervention de la part du développeur.

Une lentille magique est une vue dans un monde virtuel parallèle au monde virtuel principal. Quand le développeur crée des objets graphiques, il peut préciser s'ils sont visibles dans le monde virtuel principal ou dans le monde associé à une lentille. Les objets visibles dans les lentilles sont des objets graphiques standard, qui peuvent être stockés et affichés par le client comme des objets classiques. Ce type de lentille permet de transformer la représentation graphique des objets concernés mais ne permet pas d'y appliquer des calculs spécifiques.

8 Conclusion et perspectives

Nous avons présenté de nouvelles aides contextuelles, les couches de contexte et d'historique, qui sont destinées à faciliter l'orientation et la navigation des utilisateurs dans les grands espaces d'information disponibles dans les ZUIs. La superposition de couches transparentes permet d'intégrer le focus et le contexte dans la même vue. Ces techniques seront prochainement évaluées plus en détail afin de mieux comprendre comment elles sont utilisées.

Les hiérarchies présentées fournissent un nouveau moyen de combiner le focus et le contexte dans les ZUIs. Elles aident les utilisateurs à ne pas se perdre dans l'espace d'information et elles les renseignent sur les informations disponibles (mais cachées à un niveau de zoom donné). Une évaluation a montré l'utilité de cette technique de représentation. Des évaluations complémentaires seront nécessaires pour examiner comment les utilisateurs se servent des hiérarchies et des couches transparentes quand ces deux aides sont conjointement disponibles.

Nous proposons également un nouveau type de menu appelé Control menu, destiné à faciliter la maîtrise de ces ZUIs complexes. Nous suggérons que l'utilisation de ces menus ne se limite pas aux

ZUIs : leur usage est potentiellement général et ces menus pourraient être appliqués à d'autres types d'interfaces, comme par exemple des traitements de texte ou des logiciels de présentation.

Nous espérons que la combinaison de ces nouvelles aides de navigation et de ce nouvel interacteur pourra offrir aux utilisateurs un meilleur contrôle sur une meilleure interface. Une évaluation plus poussée nécessitera d'intégrer ces aides de navigation et les Control menus dans des logiciels usuels afin de comparer ces nouvelles techniques avec l'existant. Il serait enfin souhaitable d'effectuer une analyse plus fine des propriétés des Control menus en appliquant la loi de Fitts de manière plus approfondie.

9 Remerciements

Nous souhaitons remercier nos collègues à Infobiogen (en particulier pour leur assistance pendant l'évaluation) ainsi que nos collègues de l'École des Mines de Nantes (avec lesquels nous avons eu de fructueux échanges lors de l'utilisation de Zomit pour la réalisation d'une bibliothèque virtuelle). Nous tenons également à remercier Laurence Samarcq pour sa relecture attentive ainsi que nos relecteurs anonymes dont les remarques et suggestions ont permis une amélioration substantielle de la qualité de cet article. Ce travail a bénéficié du soutien de l'Union européenne sous le contrat BIO4-CT96-0346 et de France Télécom R&D sous le contrat 97 754 21.

Bibliographie

[Beaudouin-Lafon 2000] Beaudouin-Lafon M. « Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces ». *CHI 2000*. ACM, 2000. pp. 447–453.

[Bederson et al. 1996] Bederson B.B., Hollan J.D., Perlin K., Meyer J., Bacon D. et Furnas G. « Pad++: A Zoomable Graphical Sketchpad For Exploring Alternate Interface Physics ». *J. Vis. Lang. Comput.*, vol. 7, (1996), pp. 3–32.

[Callahan et al. 1988] Callahan J., Hopkins D., Weiser M. et Shneiderman B. « An empirical comparison of pie vs. linear menus ». *CHI '88*. ACM, 1988. pp. 95–100.

[Conklin et Begeman 1988] Conklin J. et Begeman M.L. « gIBIS: A Hypertext Tool for Exploratory Policy Discussion ». *ACM Transactions on Office Information Systems*, vol. 6, n° 4, (1988), pp. 303–331. Selected Papers from CSCW '88.

[Cox et al. 1998] Cox D.A., Chugh J.S., Gutwin C. et Greenberg S. « The Usability of Transparent Overview Layers ». *CHI '98 Summary*. ACM, 1998. pp. 301–302.

[Fekete et Plaisant 1999] Fekete J.D. et Plaisant C. « Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization ». *CHI '99*. ACM, 1999. pp. 512–519.

[Furnas 1986] Furnas G.W. « Generalized Fisheye Views ». *CHI '86*. ACM, 1986. pp. 16–23.

[Furnas et Bederson 1995] Furnas G.W. et Bederson B.B. « Space-Scale Diagrams: Understanding Multiscale Interfaces ». *CHI '95*. ACM, 1995. pp. 234–241.

[Furnas et Zhang 1998] Furnas G.W. et Zhang X. « MuSE: A Multi-Scale Editor ». *UIST '98*. ACM, 1998. pp. 107–116.

[Harrison et al. 1995a] Harrison B.L., Ishii H., Vicente K.J. et Buxton W.A.S. « Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention ». *CHI '95*. ACM, 1995a. pp. 317–324.

[Harrison et al. 1995b] Harrison B.L., Kurtenbach G. et Vicente K.J. « An Experimental Evaluation of Transparent User Interface Tools and Information Content ». *UIST '95*. ACM, 1995b. pp. 81–90.

[Hightower et al. 1998] Hightower R.R., Ring L.T., Helfman J.I., Bederson B.B. et Hollan J.D. « PadPrints: Graphical Multiscale Web Histories ». *UIST '98*. ACM, 1998. pp. 121–122.

[Hopkins 1991] Hopkins D. « The Design and Implementation of Pie Menus ». *Dr. Dobb's Journal of Software Tools*, vol. 16, n° 12, (1991), pp. 16–26,94.

[Jacob et Sibert 1992] Jacob R.J.K. et Sibert L.E. « The Perceptual Structure of Multidimensional Input Device Selection ». *CHI '92*. ACM, 1992. pp. 211–218.

[Kurtenbach et Buxton 1994] Kurtenbach G. et Buxton W. « User Learning and Performance with Marking Menus ». *CHI '94*. ACM, 1994. pp. 258–264.

[Lecolinet et al. 2001] Lecolinet E., Fekete J.D. et Pook S. « Bibliothèques : comparaisons entre le réel et le virtuel en 3D, 2D zoomable et 2D arborescent ». *ASTI 2001*. Association Française des Sciences et Technologies de l'Information, 2001. pp. 24–25.

[MacKenzie 1995] MacKenzie I.S. « Movement Time Prediction in Human-Computer Interfaces ». R.M. Baecker, W.A.S. Buxton, J. Grudin et S. Greenberg (Eds.), *Readings in human-computer interaction*. Kaufmann, 2 ed., 1995. pp. 483–493.

[Pook et al. 2000] Pook S., Lecolinet E., Vaysseix G. et Barillot E. « Control Menus: Execution and Control in a Single Interactor ». *CHI 2000 Extended Abstracts*. ACM, 2000. pp. 263–264.

[Pook et al. 1998] Pook S., Vaysseix G. et Barillot E. « Zomit: biological data visualization and browsing ». *Bioinformatics*, vol. 14, n° 9, (1998), pp. 807–814.

[Raskin 2000] Raskin J. *The Humane Interface*. Addison-Wesley, 2000.

[Robertson et Mackinlay 1993] Robertson G.G. et Mackinlay J.D. « The Document Lens ». *UIST '93*. ACM, 1993. pp. 101–108.

[Stone et al. 1994] Stone M.C., Fishkin K. et Bier E.A. « The Movable Filter as a User Interface Tool ». *CHI '94*. ACM, 1994. pp. 306–312.

[Viara et al. 1999] Viara E., Barillot E. et Vaysseix G. « The EyeDB OODBMS ». *International Database Engineering and Applications Symposium*. IEEE, 1999. pp. 390–402.