Treemaps Zoomables : Techniques d'Interaction Multi-Échelles pour les Treemaps

Renaud Blanch

Laboratoire d'Informatique de Grenoble*
385, rue de la Bibliothèque, B.P. 53
F-38041 Grenoble cedex 9, France
blanch@imag.fr

Éric Lecolinet

École Nationale Supérieure†
des Télécommunications
46, rue Barrault
F-75634 Paris cedex 13, France
elc@enst.fr

RÉSUMÉ
Visualiser des grands arbres est un problème difficile auquel des réponses efficaces (comme les *treemaps*) ont été proposées. Cependant, l'interaction avec de telles représentations est, elle aussi, problématique et les solutions actuelles ne sont pas satisfaisantes, en particulier pour des jeux de données de grande taille. Nous proposons un ensemble de nouvelles techniques d'interaction, cohérentes entre-elles, pour interagir avec les *treemaps* rendus continûment zoomables. Ces techniques tirent parti de la structure des données pour guider la navigation et enrichir ainsi les interactions traditionelles tant des *treemaps* que des interfaces zoomables.

MOTS CLÉS : Interaction multi-échelles, navigation guidée par le contenu, treemaps zoomables, visualisation d'information.

ABSTRACT
Some efficient visualizations (such as treemaps) have been proposed for trees, but the interaction they provide to explore and acces data is often poor, especially for very large trees. We have designed a *consistent set of navigation techniques* that makes it possible to use treemaps as zoomable interfaces. We introduce *structure-aware navigation*, the property of using the structure of the displayed information to guide navigation, property that our interaction techniques share.

CATEGORIES AND SUBJECT DESCRIPTORS: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces — *Graphical user interfaces, Interaction styles.*

GENERAL TERMS: Design, Human factors

KEYWORDS: Information visualization, multi-scale interaction, structure-aware navigation, zoomable treemaps.

*CNRS, INPG, INRIA, UJF, UPMF / UMR 5217.
†ENST (GET) - INFRES, CNRS LTCI / UMR 5141.

IHM '07, 13-15 Novembre 2007, Paris, France

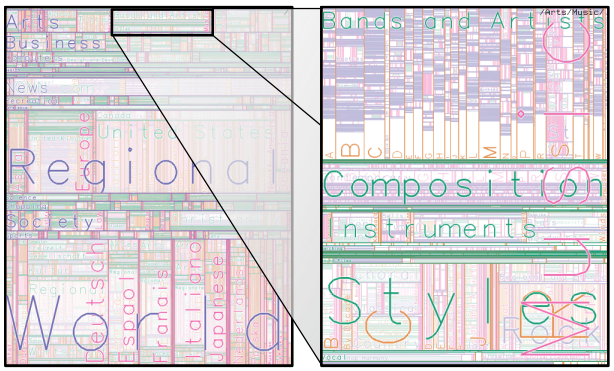

Figure 1 : Arbre affiché comme un *treemap* zoomable. (gauche) 694,986 catégories de l'ODP [1], (droite) vue donnant les détails de la catégorie *Arts/Music*.

INTRODUCTION

Les jeux de données de grande taille ayant une structure hiérarchique sont très répandus, et il est fréquent d'avoir à naviguer dans ces arbres. La recherche d'un fichier sur un disque dur, ou d'un site Web parmi un répertoire de sites en sont des exemples courants. Ces arbres deviennent de plus en plus grands à mesure des progrès techniques et des capacités accrues de stockage. Par exemple, l'*Open Directory Project* (ODP) [1] contient à ce jour plus de 694 986 catégories de sites Web qui sont réparties sur 13 niveaux. Autre exemple, le répertoire de travail de l'auteur comporte environ 172 000 fichiers répartis sur 14 niveaux.

Les représentations classiques des arbres (du type nœuds-liens) ont pour défaut d'entraîner un gaspillage important de la place disponible sur l'écran [19, 21]. C'est pour cette raison qu'ont été proposées des techniques plus avancées comme par exemple les *treemaps* [26]. Les *treemaps* sont une représentation bidimensionnelle des arbres où les noeuds sont affichés sous forme de rectangles emboîtés. La structure de l'arbre est donc portée par l'agencement de ces rectangles, les relations de parenté entre les noeuds étant visuellement représentées par l'inclusion des rectangles correspondants les uns dans les autres. La taille de chaque rectangle permet de plus de représenter un attribut quantitatif associé au noeud correspondant, d'autres attributs pouvant être codés par la couleur ou les autres carac-

téristiques graphiques. Par exemple, la Figure 1 (gauche) visualise les catégories de l'ODP à l'aide d'un *treemap*. Sur cette représentation, la taille de chaque élément indique le nombre de sites Web que contient la catégorie.

Beaucoup d'efforts ont été déployés ces dernières années pour améliorer les propriétés graphiques des *treemaps* (e.g. [27, 11, 5]). Par contre, les techniques d'interaction destinées à leur manipulation ont fait l'objet de bien moins d'attention. C'est à ce point que nous nous sommes intéressés. De par leur construction récursive qui en fait des objets « naturellement » multi-échelles, les *treemaps* sont de bons candidats pour être utilisés en tant qu'*interfaces zoomables* (ZUIs) [20, 6]. Nous proposons le concept de « *treemaps* zoomables », c'est-à-dire un ensemble de techniques d'interaction nouvelles spécifiquement adaptées au parcours de grands jeux de données arborescents représentés au moyen de *treemaps* rendus continûment zoomables.

Nous présentons tout d'abord les techniques d'interaction proposées pour la navigation au sein des *treemaps* zoomables (ZTMs). Des détails concernant leur réalisation logicielle sont ensuite donnés. Enfin, nous confrontons nos contributions à l'état de l'art.

INTERACTIONS POUR LES TREEMAPS ZOOMABLES

Les techniques d'interaction usuelles utilisées pour les *treemaps* deviennent impraticables pour des jeux de données de grandes tailles. De même, considérer les *treemaps* comme de simples espaces zoomables et leur appliquer les techniques d'interaction des ZUIs n'est pas suffisant pour explorer efficacement les données. C'est pourquoi nous proposons deux modes d'interaction pour les ZTMs. Le premier, inspiré de celui utilisé par les *treemaps*, permet de naviguer dans l'arbre en passant d'un nœud à un autre. Le second, inspiré des ZUIs autorise des mouvements libres de l'utilisateur au sein de l'espace d'information représentant l'arbre. Ces modes d'interaction sont dénommés respectivement « interaction discrète » et « interaction continue ». Ces deux modes peuvent être intégrés et multiplexés harmonieusement.

Les Treemaps Revisités : Interaction Discrète

Les différentes réalisations des *treemaps* disponibles (e.g. Discovery [4]) proposent une interaction comparable à celle de la version originale proposée par le *Human-Computer Interaction Lab* de l'Université du Maryland [22], prise ici pour référence. La navigation dans l'arbre repose sur la possibilité de sélectionner chaque nœud individuellement en pointant dans l'espace réservé pour afficher son étiquette. Ainsi, la sélection d'un nœud de la Figure 2 gauche permet d'obtenir une nouvelle vue, montrée sur la Figure 2 droite, ne comportant que le contenu de ce nœud. Aucune transition entre les deux vues n'est proposée.

Comme le montre la Figure 2, le cadre présent autour de chaque nœud occupe une hauteur constante et la place disponible à l'intérieur d'un nœud pour représenter ses fils devient vite insuffisante. La seule manière de pouvoir représenter tous les niveaux d'un arbre très profond est de ne pas réserver de place pour l'étiquette. C'est ainsi qu'ont procédé Fekete et Plaisant [13] pour afficher des arbres de l'ordre du million de nœuds.

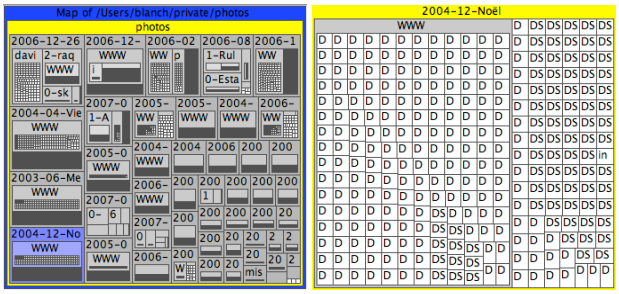

Figure 2 : Arbre affiché à l'aide du logiciel *Treemap 4*. (gauche) vue globale, (droite) détail d'un élément.

Un inconvénient de cette solution est que les étiquettes ne disposent plus d'une place réservée et peuvent alors se superposer. Nous verrons dans la partie concernant la réalisation de notre système comment apporter une solution à ce nouveau problème de lisibilité. Un autre inconvénient concerne directement l'interaction : on ne peut plus accéder par pointage à tous les nœuds affichés. En effet, de par l'emboîtement, pointer un pixel particulier de la visualisation ne permet pas de désigner de manière univoque un nœud particulier : chaque point appartient au plus profond des nœuds se situant à cet endroit, mais aussi à son père et à tous ces ancêtres. Un point caractérise ainsi en fait une *branche* de l'arbre et non un *nœud*.

Nous avons conçu nos techniques d'interaction en fonction de cette contrainte imposée par la visualisation. Ces techniques permettent de naviguer tant en profondeur, qu'en largeur dans l'arbre, et même d'accéder directement à tout nœud visible de l'arbre.

Navigation en profondeur. La navigation « en profondeur » permet de naviguer en montant et en descendant dans l'arbre. La position du curseur spécifiant une branche de l'arbre, un clic souris permet de descendre d'un niveau le long de cette branche (à l'inverse un clic droit permettant de remonter). Il faut cependant définir un nœud courant pour spécifier complètement l'interaction, ce nœud établissant par rapport à quel niveau dans l'arbre on descend (resp. on monte) d'un étage. Nous définissons le *nœud courant* par le nœud qui occupe tout l'écran.

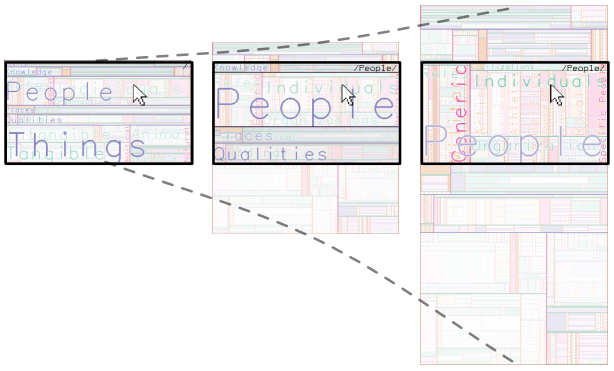

Figure 3 : Navigation en profondeur. Descente de la racine au premier niveau *People*.

Figure 4 : Navigation en largeur.
Un clic dans la marge « tourne la page » vers le prochain nœud.

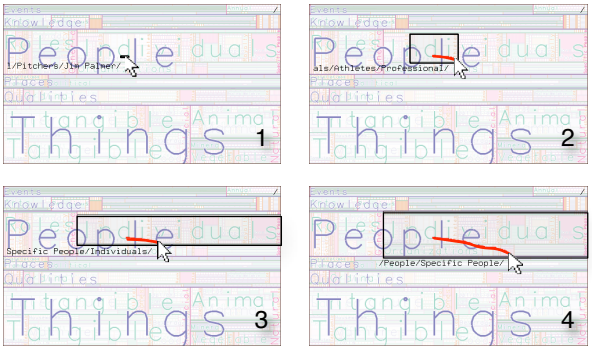

Figure 5 : Navigation en accès direct.
Une trace permet de sélectionner sans ambiguïté un nœud à une profondeur quelconque.

La Figure 3 représente cette interaction : à gauche, le nœud courant est la racine de l'arbre qui occupe tout l'écran (cadre noir). Un clic à la position du curseur indique qu'il faut descendre dans la branche située à cet endroit (dont le niveau suivant, *People*, est visible). Pour réduire la désorientation des utilisateurs au cours de la navigation, nous proposons, contrairement aux autres réalisations existantes, une transition animée (Figure 3 centre) pour amener au nouveau nœud courant (Figure 3 droite).

Navigation en largeur. La navigation en profondeur n'est pas suffisante pour certaines explorations. On peut avoir besoin de parcourir tous les fils d'un nœud situés sur un même niveau (tous les fichiers contenus dans un répertoire, tous les éléments d'une catégorie ...) Dans ce cas, la navigation en profondeur oblige à remonter au père, puis à redescendre au fils suivant pour passer d'un fils à un autre. La navigation « en largeur » permet de passer d'un élément à ses frères sans avoir à repasser par leur père.

Pour permettre cette interaction, une marge entoure la visualisation (en gris sur la Figure 4). Un clic dans cette marge (Figure 4 gauche) déclenche une animation (Figure 4 centre) qui provoque l'affichage du frère du nœud précédemment observé (Figure 4 droite). Cette animation rappelle l'action — naturelle et bien connue — de tourner les pages d'un livre. La vitesse d'exécution de cette animation est choisie suffisamment rapide pour permettre d'enchaîner plusieurs clics successifs dans la marge et de « feuilleter » ainsi rapidement un niveau de l'arbre .

Navigation en accès direct. Les techniques précédentes ne permettent que de naviguer de proche en proche dans l'arbre, en montant ou descendant au plus d'un niveau à la fois. Comme on l'a vu, cette limitation tient au fait qu'un point de la visualisation ne peut pas être associé à un élément unique de l'arbre mais seulement à une de ses branches. La représentation de l'arbre par un *treemap* donne cependant un accès visuel à plus d'un niveau à la fois et il est serait donc naturel de pouvoir accéder directement à n'importe quel nœud visible. Pour permettre cet « accès direct », nous proposons une nouvelle interaction basée sur le geste.

Si un point de la visualisation ne permet de désigner qu'une branche, il est en revanche possible d'extraire d'une trace une information plus spécifique. Comme l'illustre la Figure 5, le *plus petit nœud contenant la trace* possède de bonnes propriétés pour l'interaction. Au début de la trace (enfoncement du bouton), il s'agit de la feuille de l'arbre située à l'extrémité de la branche se situant sous le curseur (Figure 5.1). L'interaction se poursuit lorsque l'utilisateur déplace la souris en maintenant le bouton enfoncé. La trace correspondante est alors dessinée continûment. Si elle sort du nœud initialement sélectionné (Figure 5.2), le plus petit nœud qui la contient se trouve, par construction du *treemap*, être un des ancêtres du nœud précédemment sélectionné. La sélection « remonte » ainsi dans l'arbre au cours de l'interaction de telle sorte que ce soit le nœud du niveau supérieur incluant la nouvelle trace (Figure 5.3–4) qui devienne sélectionné. Finalement, au moment où le bouton est relâché, une animation amène le nœud sélectionné au premier plan, de la même manière que pour les techniques déjà présentées.

La position du clic initial spécifie donc une branche de l'arbre, et la trace permet de spécifier, de manière absolue, la profondeur que l'on veut atteindre le long de cette branche. Cette interaction est en fait une technique qui repose sur le paradigme du « franchissement » [2], lequel a été récemment proposé comme une alternative aux techniques d'activation classiques [3, 12]. Il suffit en effet que la trace traverse la frontière séparant deux frères dans l'arbre pour sélectionner leur ancêtre commun. L'utilisateur n'a pas à comprendre comment fonctionne cette interaction : un trait tracé rapidement suivant approximativement la diagonale d'un nœud aura l'effet désiré de le sélectionner.

En utilisant cette interaction, il est possible de franchir un nombre arbitraire de niveaux dans l'arbre pour accéder directement à un nœud quelconque, à condition que celui-ci soit accessible visuellement. De plus, la marge utilisée pour la navigation en largeur (Figure 4) trouve une nouvelle utilité ici : elle permet d'effectuer une trace qui se termine à l'extérieur du nœud courant, et ce faisant, traverse son bord. L'algorithme de sélection remonte alors naturellement au parent du nœud courant. La même technique ne permet donc pas seulement de descendre dans l'arbre, mais aussi de remonter dans sa hiérarchie tout en utilisant une technique d'interaction gestuelle intégrée aux autres techniques d'interaction.

Les Interfaces Zoomables Revisitées : Interaction Continue

Les techniques décrites précédemment ont une limitation commune : elles ne permettent d'accéder qu'aux nœuds ayant une représentation suffisamment étendue pour que l'on puisse interagir avec. Dans des arbres larges et peu équilibrés, ou lorsqu'on utilise un écran de petite taille, il peut arriver que la place disponible pour représenter certains nœuds soit très petite du fait du « poids » beaucoup plus important de leurs frères. Dans ce cas, l'utilisation d'une navigation qui ne soit pas dirigée par la sélection d'éléments s'avère indispensable.

Le fait que les *treemaps* soient des objets intrinsèquement multi-échelles nous a amené à les considérer comme des objets zoomables. Cependant, la métaphore de navigation libre proposée pour les ZUIs (contrôle continu de la position — *pan* — et de l'altitude — *zoom* —, parfois dénommée $2D\frac{1}{2}$) [20, 6] n'est pas pleinement adaptée aux *treemaps*. En effet, les algorithmes de placement des *treemaps* peuvent produire des rectangles aux proportions très étirées, ce problème s'aggravant avec la profondeur de l'arbre puisque l'étirement se propage d'un niveau à ses fils. Si on applique un zoom géométrique pur en un point, on ne pourra alors afficher les nœuds traversés très étirés que partiellement, ce qui rend difficile leur interprétation et rend la navigation moins aisée. La Figure 6 gauche montre l'effet résultant : au cours du zoom, on se retrouve dans une configuration où il n'est plus possible d'afficher les étiquettes et l'utilisateur est alors complètement perdu.

Snap-Zoom. C'est pour pallier cette limitation que nous avons introduit la technique du « *snap-zoom* » (zoom collant). Celle-ci dissocie les échelles de l'axe horizontal et de l'axe vertical au cours du zoom afin que les proportions des rectangles représentant les nœuds soient optimalement adaptées aux proportions de l'écran (Figure 6 droite).

Ainsi, l'utilisateur utilise la molette de la souris pour zoomer ou dézoomer à l'endroit du curseur, et le *snap-zoom* dilate ou contracte la visualisation en la déformant régulièrement et graduellement pour que les niveaux traversés successivement prennent la taille et les proportions de l'écran quelles que soient leurs tailles et leurs proportions initiales. Si le nœud situé sous la souris est très allongé horizontalement, le zoom sera appliqué principalement sur l'axe vertical jusqu'à ce que le nœud atteigne la taille de l'écran. Si au contraire il est étiré verticalement, le zoom sera appliqué sur l'axe horizontal. Cela permet de préserver une structure compréhensible et des étiquettes lisibles au cours de la progression dans les niveaux de l'arbre. L'algorithme et les formules spécifiant la manière dont les échelles horizontale et verticale évoluent indépendamment sont détaillés dans la partie présentant la réalisation des *treemaps* zoomables.

Les changements d'échelle affectant globalement le *treemap*, les tailles relatives des rectangles restent toujours comparables. Notre technique conserve donc une propriété fondamentale des *treemaps* : l'utilisation de la surface pour véhiculer une information quantitative. Les techniques de déformation à la *fisheye* [14] ne permettent pas de préserver cette propriété.

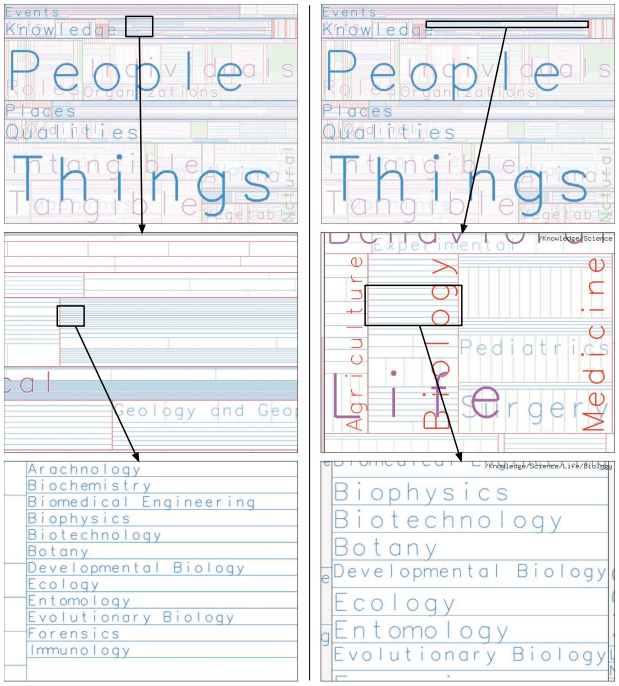

Figure 6 : *Snap-zoom.*
(gauche) zoom géométrique, (droite) *snap-zoom*.

Une bonne propriété du *snap-zoom* est que même si c'est le système qui détermine précisément les facteurs de zoom pour les deux axes, l'utilisateur les manipule directement par le biais d'une seule variable qui est le produit de ces deux facteurs. Cette variable conserve un sens physique : c'est le facteur d'échelle global de la visualisation. L'utilisateur garde donc un contrôle complet sur la variable importante de l'interaction : le facteur de zoom. Celui-ci peut être concrètement manipulé par la molette de la souris pour permettre une interaction fluide et un contrôle fin par l'utilisateur.

Multiplexage des Modes Discret et Continu

La présentation des techniques d'interaction discrètes n'a pas détaillé la nature des animations utilisées pour passer d'un nœud courant à un autre. En fait, les animations de ces interactions présentées plus haut utilisent aussi des facteurs de zoom différents selon les deux axes pour amener le nœud cible à remplir l'écran, quelles que soient ses caractéristiques géométriques initiales. Toutes les techniques utilisent en fait la même fonction de zoom, mais dans le cas des interactions discrètes, les pas de l'animation sont déclenchés par le programme alors que dans le cas du *snap-zoom*, c'est l'utilisateur qui contrôle son évolution. Cette dernière interaction peut ainsi être vue comme une animation dont le contrôle temporel est donné à l'utilisateur par le biais de la dimension d'échelle.

Cette unification de l'animation entre les modes discret et continu permet de passer d'une interaction à l'autre sans effort de la part de l'utilisateur. Elle nous a aussi permis d'explorer un nouveau type de combinaison de ces techniques par le biais d'un menu contextuel inspiré des *marking menus* [18] et des *control menus* [23].

(a) (b) (c) (d)

Figure 7 : *Zoom menu* en mode discret : (a, b) la cible est sélectionnée, puis (c, d) l'animation est jouée.

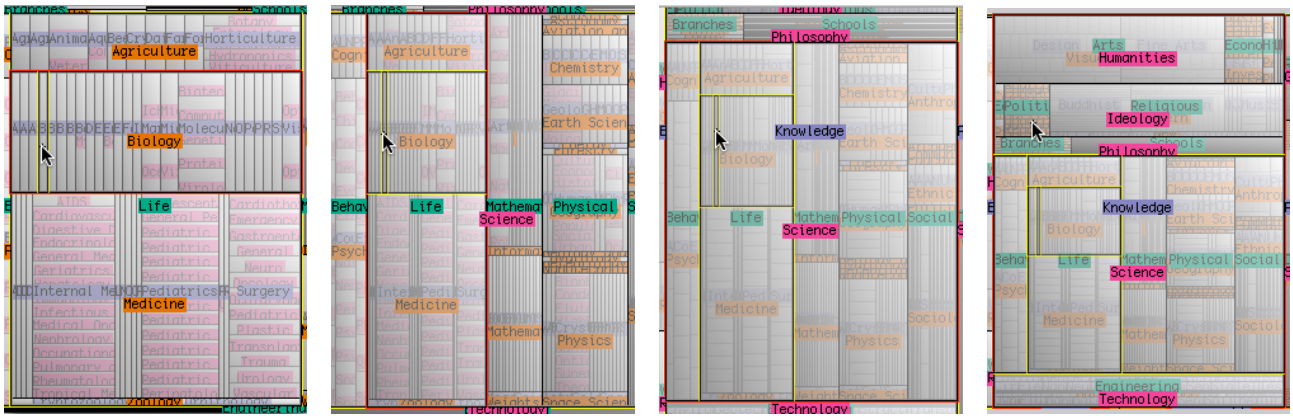

Figure 8 : *Zoom menu* en mode continu : l'animation est contrôlée directement par l'interaction.

Zoom Menu. Le *zoom menu* est un menu contextuel qui, dans sa forme simple, présente à l'utilisateur une liste comportant les étiquettes des différents nœuds de la branche spécifiée par le point où le curseur se situe lors de l'enfoncement du bouton (Figure 7.a). La liste est positionnée telle que le nœud courant, c'est-à-dire celui qui occupe tout l'écran (la catégorie *Life* dans notre illustration), se situe sous le curseur. Au-dessus se trouvent les nœuds rencontrés en partant du nœud courant et en remontant vers la racine de l'arbre (*Science*, *Knowledge* et enfin la racine de l'arbre *Categories*). Au-dessous se trouvent les nœuds de la branche trouvés en descendant vers les feuilles (*Biology* et enfin la feuille *Biochemistry*). Lorsque le curseur est relâché, l'élément du menu situé sous le curseur est sélectionné (Figure 7.b). Le nœud correspondant devient alors le nœud courant grâce à une animation qui l'amène en plein écran (Figure 7.c–d).

Le *zoom menu* combine avec cette interaction discrète un second mode d'interaction qui, lui, est continu. Pour ce faire, le menu n'apparaît en fait que si le bouton de la souris est enfoncé et que le curseur n'est pas déplacé pendant un court instant, à la manière du mode novice des *marking menus* [18]. Si le curseur est déplacé directement après le clic, c'est le second mode de l'interaction qui est utilisé. Dans ce mode, les déplacements du curseur contrôlent continûment le zoom du document. Seul la composante verticale du déplacement est prise en compte : un déplacement vers le bas zoome dans le document, alors qu'un déplacement vers le haut dézoome. Ainsi, sur la Figure 8, l'utilisateur remonte dans l'arbre par un simple geste dirigé vers le haut.

Cette interaction n'est pas compatible avec l'interaction d'accès direct présentée plus haut qui repose aussi sur le geste. Cependant, le mode discret du menu fournit cet accès direct par le biais d'un contrôle différent. De même, le mode continu du *zoom menu* fournit le même accès que le *snap-zoom* mais avec un contrôle différent. Nous le proposons donc comme un moyen alternatif de multiplexer ces deux modes d'interaction. Contrairement aux techniques précédentes, le multiplexage n'a pas lieu au niveau du dispositif physique (bouton pour la trace, molette pour le *snap-zoom*), mais sur le plan temporel, ce qui permet d'utiliser le même périphérique d'interaction pour les deux modes. Le *zoom menu* est donc particulièrement adapté à une interaction sur un dispositif n'offrant pas beaucoup de possibilités physiques comme un PDA.

Il faut noter que l'intégration des deux modes au sein d'un même interacteur est réalisée de manière cohérente : dans les deux cas un déplacement vers le haut remonte dans l'arbre, alors qu'un déplacement vers le bas permet de s'y enfoncer. Cette compatibilité des actions permet de passer d'un mode à l'autre sans effort.

RÉALISATION

Le prototype utilisé pour mettre au point nos techniques d'interaction a été réalisé à l'aide du langage C++ en utilisant la bibliothèque graphique OpenGL pour bénéficier de l'accélération matérielle du rendu graphique. La bibliothèque GLUT a été utilisée pour gérer la communication avec le système de fenêtrage et ses capacités de rendu de texte.

Nous détaillons dans cette partie quelques points précis de la réalisation du système. Les algorithmes sous-jacents aux diverses techniques d'interaction sont tous d'abord présentés. Les techniques utilisées pour permettre de garder le rendu lisible sont ensuite détaillées. Enfin, les optimisations particulières utilisées pour préserver l'interactivité du système malgré la grande quantité d'information manipulée sont expliquées.

Interactions

Deux problèmes ont dû être résolus pour permetter la combinaison cohérente des différentes techniques d'interaction. Tout d'abord, les interactions discrètes ont été jusqu'à présent décrites en terme de déplacement par rapport à un nœud courant, celui qui est « plein écran ». Cependant, les interactions continues peuvent laisser le système dans un état où il n'y a pas de nœud répondant à cette définition. Il faut la raffiner, et préciser comment est choisi la cible des interactions. Un second élément, pour l'instant passé sous silence, est la manière dont le système répartit une quantité de zoom donnée selon les deux axes pour afficher au mieux les nœuds traversés au cours du zoom.

Détermination de la cible des interactions. On peut définir le nœud courant comme le nœud étant le plus en « plein écran ». Si on cherche ce nœud sous la souris, le problème est simplifié : on a vu qu'un point spécifie une branche. Nous pouvons alors chercher notre élément le long de cette branche. Par ailleurs, de par la construction du *treemap*, la taille des nœuds décroît lorsqu'on parcourt les branches depuis la racine vers les feuilles. Il existe donc (modulo les cas particuliers en haut et en bas de la branche) un couple de nœuds se succèdant sur cette branche (un père P et son fils F) tel que P ne soit pas complètement inclus dans la fenêtre et F le soit strictement. Ces deux nœuds sont ceux de la branche qui ont la taille la plus proche de la fenêtre. Ce sont les meilleurs candidats pour définir un nœud courant.

Pour les interactions qui permettent de naviguer en profondeur, la cible est choisie en fonction du sens dans lequel elles se dirigent : si elles descendent (resp. montent) dans l'arbre, la cible est F (resp. P), le nœud le plus proche n'ayant pas encore atteint la taille de l'écran (resp. étant plus grand que l'écran). Pour les autres interactions, on choisit par convention P comme nœud courant, les déplacement se faisant alors relativement à lui.

Réalisation du zoom. Une fois désignée la cible, il faut choisir le facteur de zoom à effectuer dans sa direction. Dans le cas d'une animation, il s'agit d'un pas constant, dans celui d'une interaction, il est fixé par l'amplitude su mouvement qui la contrôle. Ce facteur de zoom de la surface, δs^2, doit simplement respecter $\delta s^2 > 1$ pour un zoom avant et $\delta s^2 < 1$ pour un zoom arrière. L'algorithme va alors répartir ce facteur de zoom global en deux facteurs, l'un pour l'axe horizontal, δs_x, et l'autre pour l'axe vertical, δs_y. La contrainte à respecter pour que δs^2 représente bien le zoom du document est que $\delta s_x \times \delta s_y = \delta s^2$.

Afin de décider la distribution de ce facteur de zoom suivant ses deux composantes, on calcule les facteurs de zoom globaux qu'il faudrait appliquer pour que la cible atteigne la taille de l'écran, soient : $\Delta s_x = W/w$ et $\Delta s_y = H/h$ où W et H (resp. w et h) sont les largeur et hauteur de l'écran (resp. de la cible). δs^2 est alors séparé selon ses deux composantes selon les mêmes proportions que le facteur de zoom global $\Delta s^2 = \Delta s_x \times \Delta s_y$. Cependant, le zoom à vitesse constante impliquant une progression géométrique de l'échelle, et non arithmétique, les proportions doivent être considérées pour l'indice de zoom, c'est-à-dire le logarithme de l'échelle [16]. En prenant l'indice de zoom $z = \log(s)$, on obtient :

$$\delta z_x = \frac{\delta z^2}{\Delta z^2}\Delta z_x, \;\; \delta z_y = \frac{\delta z^2}{\Delta z^2}\Delta z_y \text{ et donc :}$$

$$\delta s_x = \Delta {s_x}^t, \;\; \delta s_y = \Delta {s_y}^t, \text{ avec } t = \frac{\log \delta s^2}{\log \Delta s^2}.$$

Rendu

Le principal défi à relever pour représenter un telle quantité d'information est de minimiser l'espace perdu tout en évitant au maximum la surcharge visuelle. Pour ce faire, les nœuds sont dessinés en partant du fond afin que les hauts niveaux ne soient pas occultés par les détails plus petits. Quatre couleurs sont utilisées à tour de rôle pour représenter les divers niveaux. Ces couleurs appartiennent à une palette conçue pour que chacune d'entre-elles se différencie aisément des autres [10]. Ceci permet que les étiquettes d'un niveau se détachent de celles du niveau inférieur, même si elles sont superposées.

Les niveaux sont rendus d'autant moins visibles qu'ils sont profonds dans l'arborescence grâce à une atténuation. Celle-ci est adaptée de telle manière qu'au cours de l'exploration en profondeur, les détails se révèlent à mesure qu'ils deviennent pertinents pour le niveau d'observation courant. De même, les niveaux supérieurs de l'arbre situés au-dessus du nœud courant ne sont pas affichés pour ne pas le masquer. Pour ce faire, les niveaux sont répartis spatialement en trois dimensions sur un axe orthogonal à l'écran suivant leur profondeur dans l'arbre. La vue se déplace selon cet axe lorsque l'utilisateur zoome, ce qui permet d'éliminer les niveaux supérieurs grâce à un calcul de visibilité délégué à la bibliothèque OpenGL. Comme la projection de l'espace 3D sur l'écran 2D est orthographique, les positions des nœuds dans le plan de l'écran ne sont pas altérées par ce placement dans l'espace.

Performance

Pour obtenir des temps de rendu compatibles avec une utilisation interactive (soit au moins 10 images par secondes), trois types d'optimisation sont effectués. La plus simple concerne le rendu du texte : les étiquettes ne sont affichées que si la taille apparente du texte leur permet d'être lisible. Le rendu de texte étant assez coûteux, cette optimisation élémentaire réduit considérablement le temps de rendu.

Les autres optimisations concernent les nœuds eux-mêmes. Tout d'abord, les nœuds situés à l'extérieur de la fenêtre sont éliminés. Par construction des *treemaps,* leurs fils n'ont pas non plus besoin d'être considérés. On peut ainsi couper rapidement des branches entières en parcourant l'arbre en partant du sommet. À l'autre extrémité, les nœuds ayant une taille inférieure à un seuil donné (typiquement le pixel) ne sont pas non plus rendus et leurs descendants sont ignorés. Le seuil est adapté dynamiquement suivant la performance instantanée du système : si le rendu devient trop lent, la taille minimale est augmentée, ce qui a pour effet de couper plus haut dans l'arbre et d'éliminer davantage de détails. Au contraire, si le programme dispose d'un temps suffisant pour un meilleur rendu, le seuil est révisé pour permettre un affichage plus détaillé. Cette heuristique permet de fixer dynamiquement le compromis entre niveau de détail et interactivité. Notre prototype peut ainsi fonctionner efficacement sur des machines même modestes.

Par exemple, sur la Figure 1 gauche, l'arbre comporte 697 986 nœuds, mais l'algorithme n'affiche que les 145 841 noeuds les plus importants, puisque les autres auraient une taille inférieure au pixel. Sur cette figure, seules 235 étiquettes sont affichées, puisque les autres ne seraient pas lisibles. Ainsi, l'arbre peut être réaffiché plus de 30 fois par secondes sur une machine d'entrée de gamme, ce qui permet une interaction continue complètement fluide.

Prototype

Les techniques d'interaction décrites ci-dessus sont réalisées au sein d'un prototype dont le code source est mis à la disposition du public[1]. Un prototype précédent comportant un sous-ensemble de ces techniques a été présenté par une démonstration à la conférence UIST'06 [9].

ÉTAT DE L'ART

Interaction & Treemaps Zoomables

Une implémentation de référence des *treemaps* est proposée par le *Human-Computer Interaction Lab* de l'Université du Maryland [22]. Elle dispose d'une simple navigation en profondeur, dépourvue d'animation. Les autres implémentations disponibles ont les mêmes limitations. Ceci est sans doute dû au fait que les recherches ont jusqu'à ce jour essentiellement porté sur l'aspect visualisation, qui est en soi un challenge, plus que sur l'aspect interaction de ce type de représentation. *PhotoMesa* [5] utilise également des treemaps pour afficher et parcourir des répertoires contenant des photos. Ce logiciel permet un zoom continu mais ne conserve pas et n'utilise pas la structure hiérarchique des données pour guider l'interaction comme nous le faisons. *StepTree* [8] utilise le zoom pour animer les transitions lors de l'exploration en profondeur dans un *treemap* mais il ne permet pas d'interaction continue.

En ce qui concerne les interfaces zoomables (introduites par Perlin et Fox [20]) plusieurs techniques d'interaction ont été proposées comme les portails *(portals)*, la navigation libre *(pan-and-zoom)*, les menus pseudo-gestuels de type *control menus*. Notre apport est d'avoir relaxé la contrainte du zoom géométrique. En autorisant des facteurs d'échelle différents sur les deux axes, nos techniques sont adaptées à tous types de placement des *treemaps* et lèvent une limitation connue de cette représentation : le problème des rectangles ayant des proportions très disparates.

Par ailleurs, le fait de pouvoir accéder directement à l'aide d'un geste à des nœuds enfouis dans l'arbre est aussi une amélioration notable de l'existant. Shi et al. [25] ont évalué une technique permettant aussi cet accès direct, mais reposant sur une déformation *fisheye* de la visualisation. Une telle déformation n'est cependant pas satisfaisante car elle modifie la disposition des rectangles et perd ainsi une propriété importante des *treemaps* pour laquelle ils sont utilisés : leur capacité à représenter un attribut quantitatif à l'aide de l'espace qu'occupe un nœud. Les *Elastic Hierarchies* [28] proposent une approche complémentaire à la notre pour accéder à des nœuds intérieurs de l'arbre en ajoutant des onglets permettant de choisir le niveau auquel on veut interagir avec l'arbre. Cette approche simple mais limitée ne permet cependant pas une interaction fluide et continue contrairement à celle que nous proposons.

Visualisation de Grands Arbres

Les travaux menés sur la visualisation de grands arbres constituent un autre domaine connexe de notre cadre d'étude. Les *treemaps* ont déjà été utilisés pour afficher de très grands arbres [13], mais ces travaux se sont concentrés sur la visualisation et non sur l'interaction. Les *SpaceTrees* [21] et *revisited degree-of-interest trees* [17] sont des contributions récentes, mais elles se sont focalisées sur des représentations « nœuds-liens ». Ces deux travaux maximisent la proportion de l'écran utilisée et proposent des indices indiquant des caractéristiques synthétiques des parties de l'arbre qu'ils ne peuvent afficher. Les nœuds affichés sont choisis suivant diverses politiques : actions explicites de l'utilisateur, pertinence vis-à-vis d'une requête textuelle, degré d'intérêt calculé par le système. Ils utilisent aussi des animations pour permettre aux utilisateurs de suivre les changements de disposition de l'arbre. Cette utilisation des animations dans les *treemaps* avait été suggérée par [15], mais pas dans un usage interactif.

L'utilisation d'animations pour l'interaction avec des représentations d'arbres utilisant l'emboîtement a été proposée avec le *structural zooming* [24], mais sans conserver la disposition propre aux *treemaps*. De manière similaire, McGuffin et al. [19] proposent un algorithme permettant d'ouvrir au maximum un arbre au cours de son parcours, en utilisant des transitions animées, mais là encore, les propriétés des *treemaps* sont perdues.

CONCLUSION ET PERSPECTIVES

Les techniques d'interaction que nous avons proposées permettent de naviguer dans de très grands arbres représentés par des *treemaps*. Elles utilisent la structure de l'arbre pour guider la navigation et éviter ainsi que l'utilisateur se retrouve perdu dans cette masse de données. Elles ne contraignent cependant pas l'utilisateur dans sa navigation puisqu'il peut naviguer librement en étant simplement guidé (et non forcé) dans sa navigation. Ces techniques d'interaction s'intègrent harmonieusement ensemble et permettent de passer d'un mode dis-

[1] Le code source du prototype est téléchargeable à l'adresse : iihm.imag.fr/blanch/projects/ztm/.

cret à un mode continu sans que la transition entre ces deux modes soit discernable. Nos premiers tests utilisateurs, bien qu'informels, sont encourageants.

Nous comptons évaluer ces techniques de manière plus contrôlée, et confronter ainsi nos travaux avec l'existant. Par ailleurs, nous sommes en train d'adapter ces techniques à une game plus large de plate-formes. Aux dispositifs nomades tout d'abord, ce qui pose plusieurs problèmes : les ressources limitées en calcul des dispositifs en est un, mais nos travaux d'optimisation du rendu sont déjà pertinents. La taille limitée de l'écran est une autre variable à prendre en compte, mais là encore, pour de telles quantités d'information, l'écran est toujours trop petit, et les problèmes de place sont déjà au cœur des préoccupations de la visualisation d'information. Enfin, les dispositifs d'interaction présents sur les plates-formes mobiles sont différents de ceux de la station de travail classique. À la souris et au curseur se substitue la désignation directe à l'écran à l'aide d'un stylet. Notre *zoom menu* est un premier pas vers l'adaptation à ces petits dispositifs. À l'autre extrémité du spectre des dispositifs, l'utilisation d'une table interactive (e.g. la table magique de Bérard [7]) nécessitera aussi l'adaptation des techniques d'interaction à de nouveaux contextes (e.g. la collaboration) et à de nouvelles entrées, ouvrant ainsi de nouvelles pistes d'études.

REMERCIEMENTS

Une partie de ce travail a été réalisée lors du séjour postdoctoral de R. B. à l'ENST financé par la région Île-de-France.

BIBLIOGRAPHIE

1. Open directory project, 2006. www.dmoz.org.
2. J. Accot et S. Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proc. ACM CHI'02*, pages 73–80, 2002.
3. G. Apitz et F. Guimbretière. CrossY: a crossing-based drawing application. In *Proc. ACM UIST'04*, pages 3–12, 2004.
4. T. Baudel. From information visualization to direct manipulation: extending a generic visualization framework for the interactive editing of large datasets. In *Proc. ACM UIST'06*, pages 67–76, 2006.
5. B. B. Bederson. PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proc. ACM UIST'01*, pages 71–80, 2001.
6. B. B. Bederson et J. D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proc. ACM UIST'94*, pages 17–26, 1994.
7. F. Bérard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *Proc IEEE PROCAM'03*, 2003.
8. T. Bladh, D. A. Carr et M. Kljun. The effect of animated transitions on user navigation in 3D tree-maps. In *Proc. IEEE IV'05*, pages 297–305, 2005.
9. R. Blanch et É. Lecolinet. Navigation techniques for zoomable treemaps. In *Adj. Proc.: Demos of ACM UIST'06*, pages 49–50, 2006.
10. C. A. Brewer, 2006. www.ColorBrewer.org.
11. M. Bruls, K. Huizing et J. J. van Wijk. Squarified treemaps. In *Proc. Eurographics & IEEE TCVG Symp. on Visualization*, pages 33–42, 2000.
12. P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. ACM UIST'04*, pages 193–196, 2004.
13. J.-D. Fekete et C. Plaisant. Interactive information visualization of a million items. In *Proc. IEEE InfoVis'02*, pages 117–123, 2002.
14. G. W. Furnas. Generalized fisheye views. In *Proc. ACM CHI'86*, pages 16–23, 1986.
15. M. Ghoniem et J.-D. Fekete. Animating treemaps. In *Proc. Workshop on Treemap Implementation and Applications*. University of Maryland, 2001.
16. Y. Guiard et M. Beaudouin-Lafon. Target acquisition in multiscale electronic worlds. *Int. J. Human-Computer Studies*, 61:875–905, 2004.
17. J. Heer et S. K. Card. DOITrees revisited: Scalable, space-constrained visualization of hierarchical data. In *Proc. AVI'04*, pages 421–424, 2004.
18. G. P. Kurtenbach. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, 1993.
19. M. J. McGuffin, G. Davison et R. Balakrishnan. Expand-ahead: a space-filling strategy for browsing trees. In *Proc. IEEE InfoVis'04*, pages 119–126, 2004.
20. K. Perlin et D. Fox. Pad: an alternative approach to the computer interface. In *Proc. ACM SIGGRAPH'93*, pages 57–64, 1993.
21. C. Plaisant, J. Grosjean et B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proc. IEEE InfoVis'02*, pages 57–64, 2002.
22. C. Plaisant et B. Shneiderman. Treemap: Home page, 2006. www.cs.umd.edu/hcil/treemap/.
23. S. Pook, É. Lecolinet, G. Vaysseix et E. Barillot. Control menus: excecution and control in a single interactor. In *Ext. abs. CHI'00*, pages 263–264, 2000.
24. K. Pulo, P. Eades et M. Takatsuko. Smooth structural zooming of h-v inclusion tree layouts. In *Proc. CMV'03*, 2003.
25. K. Shi, P. Irani et B. Li. An evaluation of content browsing techniques for hierarchical space-filling visualizations. In *Proc. IEEE InfoVis'05*, pages 81–88, 2005.
26. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.
27. J. J. V. Wijk et H. van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Proc. IEEE InfoVis'99*, pages 73–78, 1999.
28. S. Zhao, M. J. McGuffin et M. H. Chignell. Elastic hierarchies: combining treemaps and node-link diagrams. In *Proc. IEEE InfoVis'05*, pages 57–64, 2005.