Metisse : un système de fenêtrage hautement configurable et utilisable au quotidien

Nicolas Roussel & Olivier Chapuis

LRI (Université Paris-Sud – CNRS) & INRIA Futurs *
Bâtiment 490, Université Paris-Sud
91405 Orsay Cedex, France
roussel | chapuis @ lri.fr

RESUME
Vingt ans après l'adoption généralisée des fenêtres superposées et de la métaphore du bureau, les systèmes de fenêtrage modernes ne diffèrent les uns des autres que sur des points de détail mineurs. Bien que de nouvelles techniques de gestion de fenêtres soient encore régulièrement proposées, peu d'entre-elles sont formellement évaluées et moins encore sont implémentées dans de réels systèmes. Dans cet article, nous présentons Metisse, un nouveau système de fenêtrage créé spécifiquement pour faciliter la conception, la mise en oeuvre et l'évaluation de nouvelles techniques de gestion de fenêtres.

MOTS CLES : Système de fenêtrage, gestion de fenêtres.

ABSTRACT
Twenty years after the general adoption of overlapping windows and the desktop metaphor, modern window systems differ mainly in minor details. While a number of innovative window management techniques have been proposed, few of them have been evaluated and fewer have made their way into real systems. In this paper, we present Metisse, a fully functional window system specifically created to facilitate the design, the implementation and the evaluation of innovative window management techniques.

CATEGORIES AND SUBJECT DESCRIPTORS: D.4.9: Window managers. H.5.2: Windowing systems.

GENERAL TERMS: Human factors, Design, Algorithms.

KEYWORDS: Windowing system, window management.

INTRODUCTION

Peu de travaux ont été consacrés à l'étude des différentes pratiques en matière de gestion de fenêtres [3]. Parmi les techniques d'interaction proposées par la communauté de recherche en IHM, rares sont celles qui ont été formellement évaluées. La plupart de ces techniques (e.g. [6, 1, 4]) ont été conçues selon une approche basse fidélité, rendant impossible toute étude de type longitudinale. Très peu ont été intégrées aux systèmes de fenêtrages les plus utilisés. Task Gallery [9] permettait bien d'intégrer des applications Windows existantes dans un environnement 3D. Toutefois, les modifications de Windows 2000 nécessaires à son fonctionnement n'ont jamais été rendues publiques.

Plusieurs travaux récents (e.g. [8, 5, 12]) ont montré que la mise en oeuvre de techniques de gestion de fenêtres en dehors du système de fenêtrage les rend inutilement complexes, inefficaces et difficiles à combiner les unes avec les autres. Plusieurs projets comme Ametista [10], Looking Glass (http://www.sun.com/software/looking_glass/) ou encore Croquet (http://www.opencroquet.org/) permettent l'intégration d'applications X Window dans des environnements graphiques expérimentaux. Mais ces environnements n'implémentent qu'une petite partie des protocoles liés à la gestion de fenêtres comme ICCCM ou EWMH, indispensables au bon fonctionnement de la plupart des applications GNOME ou KDE.

Concevoir et mettre en oeuvre un nouveau système de fenêtrage utilisable au quotidien est une tâche difficile dans laquelle peu de chercheurs en IHM sont prêts à se lancer. Dans le même temps, les systèmes existants sont soit conçus comme des boîtes noires inaccessibles aux programmeurs (Windows ou Mac OS X), soit trop complexes à modifier car reposant sur un ensemble d'extensions plus ou moins stables accumulées au fil des ans (X Window). Dans ces conditions, comment implémenter un gestionnaire de fenêtres zoomable, destiné à être utilisé sur une table interactive, ou permettant l'interaction bi-manuelle ou à plusieurs ?

Dans cet article, nous présentons Metisse, un nouveau système de fenêtrage basé sur X Window. Metisse a été conçu pour répondre à deux objectifs précis: 1) faciliter la conception et la mise en oeuvre par des chercheurs en IHM de nouvelles techniques de gestion de fenêtres ; 2) respecter les standards existants et être suffisamment robuste et performant afin de pouvoir être utilisé quotidiennement et permettre ainsi

IHM 2005, September 27-30, 2005, Toulouse, France.

* projet In Situ, Pôle Commun de Recherche en Informatique (PCRI) du plateau de Saclay, CNRS, Ecole Polytechnique, INRIA, Université Paris-Sud

l'évaluation des techniques en question. Metisse n'est pas destiné à l'étude d'un style particulier d'interaction (e.g. 3D). Il ne doit pas être vu comme un nouvel environnement graphique, mais plutôt comme un outil permettant l'exploration de nouveaux environnements.

ARCHITECTURE ET IMPLEMENTATION

Comme Ametista [10], Metisse repose sur un découplage entre le processus de rendu des fenêtres et la composition interactive des images produites. Cependant, tandis qu'Ametista n'était qu'une boîte à outils expérimentale, Metisse se veut être un système de fenêtrage complet, compatible avec les systèmes existants. Le *serveur* Metisse est un serveur X Window dérivé de Xserver (http://www.freedesktop.org/) et capable d'effectuer le rendu des fenêtres dans des zones mémoire séparées, allouées dynamiquement. La composition et l'interaction avec l'utilisateur sont gérées par une version légèrement modifiée du gestionnaire de fenêtres FVWM (http://www.fvwm.org/) associée à un visualisateur interactif : *FvwmCompositor* (Figure 1).

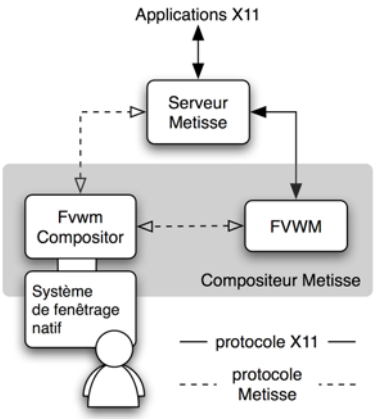

Figure 1 : Architecture de Metisse.

FvwmCompositor est une application du système de fenêtrage natif qui utilise OpenGL pour afficher les images des fenêtres gérées par le serveur et lui relaie les événements en provenance des périphériques d'entrée. FVWM prend directement en charge les opérations classiques sur les fenêtres (e.g. déplacement, redimensionnement, iconification), informant au passage le serveur des changements d'état ou de propriétés. Mais toutes ces opérations peuvent être déléguées à FvwmCompositor. L'ajout de nouvelles opérations sur les fenêtres peut se faire au niveau de FVWM, par l'intermédiaire d'un langage de script, ou dans FvwmCompositor.

Lorsqu'une application redessine l'une de ses fenêtres, le serveur notifie FvwmCompositor en lui précisant la région modifiée à l'aide d'un protocole similaire à celui utilisé par VNC [7], ou par l'intermédiaire d'une zone de mémoire partagée. D'autres notifications sont envoyées par le serveur pour indiquer notamment la création et la destruction de fenêtres et les changements de curseur. Ces notifications permettent à FvwmCompositor de tenir à jour la liste des fenêtres à afficher et des textures associées. Pour les fenêtres temporaires comme les menus *popup*, le serveur indique – dans la mesure du possible – une fenêtre de référence, ce qui permet de leur appliquer des transformations similaires.

Le serveur Metisse diffère des serveurs X traditionnels sur plusieurs points. Il ne génère jamais d'événement *Expose*, les problèmes de visibilité et d'occlusion partielle n'ayant plus de sens puisque le rendu s'effectue en mémoire. La gestion des événements provenant de la souris est également particulière. Un serveur traditionnel utilise la position du pointeur et sa connaissance du placement des fenêtres pour identifier la fenêtre devant recevoir un événement souris. Dans le cas de Metisse, puisque FvwmCompositor est libre de composer les images des fenêtres arbitrairement, les événements souris qu'il relaie vers le serveur doivent nécessairement spécifier la fenêtre concernée. Lorsque la configuration de l'écran change, ou lors d'opérations manipulant simultanément plusieurs fenêtres, FvwmCompositor doit également générer les événements *Enter* et *Leave* appropriés à la place du serveur.

Lorsqu'il reçoit un événement souris, FvwmCompositor utilise les mécanismes de sélection et de *picking* d'OpenGL pour déterminer la fenêtre sous le pointeur. Dans certains cas, comme lors d'un redimensionnement de fenêtre interactif effectué très rapidement, le pointeur peut sortir de la fenêtre concernée. Afin d'éviter ce genre de problème, les polygones correspondant aux fenêtres sont agrandis avant d'être dessinés en mode sélection. Une fois la fenêtre cible déterminée, FvwmCompositor utilise la matrice de transformation géométrique associée pour calculer les nouvelles coordonnées de l'événement à passer au serveur. Les événements clavier sont simplement relayés vers le serveur, sans modification. La politique de gestion du focus clavier est prise en charge par FVWM et peut ainsi être facilement configurée en fonction des préférences de l'utilisateur.

L'utilisation de FVWM présente plusieurs avantages par rapport à l'approche adoptée par Ametista, Looking Glass ou Croquet, dans laquelle l'application en charge de la composition des fenêtres se charge elle-aussi de toutes les opérations interactives. Nous l'avons dit, FVWM se charge de toutes les opérations classiques sur les fenêtres. Mais il sait également gérer d'autres aspects plus complexes comme la prise en compte de bureaux virtuels. Si FvwmCompositor n'applique pas de transformation particulière, Metisse se comporte donc comme un environnement X Window tout à fait standard et est directement utilisable pour un usage quotidien. Ensuite, comme les décorations ajoutées par FVWM sont elles-mêmes dessinées dans des fenêtres, elles sont automatiquement accessibles dans FvwmCompositor, à travers le serveur. À l'usage, pour le programmeur, cette solution s'est avérée bien plus pratique que la création d'équivalents OpenGL à ces bordures, barres de titre, boutons et menus qui permettent de déclencher et contrôler les opérations sur les fenêtres.

EXEMPLES D'UTILISATION

FvwmCompositor implémente plusieurs nouvelles opérations élémentaires sur les fenêtres : changement d'échelle, rotations selon les trois axes, changement de l'opacité et de la luminosi-

té de la fenêtre. FVWM permet, au moyen de scripts, d'associer ces opérations à des événements clavier/souris ou de plus haut niveau. Une combinaison de touches particulière peut ainsi déclencher la baisse de luminosité de toutes les applications exceptée celle ayant le focus clavier. Les fenêtres d'une application particulière peuvent également être rendues semi-transparentes lorsqu'elle ne sont pas manipulées.

Les nouvelles opérations peuvent être combinées avec les opérations traditionnelles. Une des fonctions offertes par Metisse réduit ainsi l'échelle d'une fenêtre puis la redimensionne afin qu'elle occupe toute la hauteur de l'écran (Figure 2). Un nouvel appel de la fonction annule la transformation, la fenêtre retrouvant ainsi son échelle et ses dimensions originales. Cette fonction permet de naviguer à échelle réduite dans un grand document, quelle que soit l'application utilisée. Il est intéressant de noter qu'elle a été conçue et développée le temps d'un trajet Chatelet-Orsay en RER (35 minutes), le code final ne représentant que quelques lignes ajoutées au fichier de configuration FVWM de Metisse.

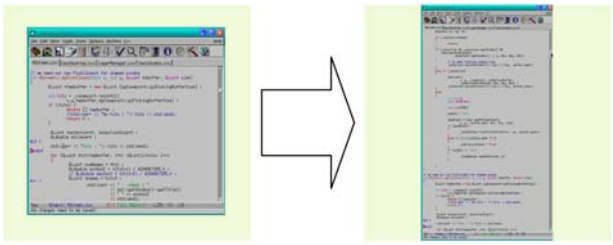

Figure 2 : Changement d'échelle accompagné d'un changement de taille.

Nous avons également implémenté une fonction qui réduit l'échelle des fenêtres lorsque celles-ci sont amenées au bord de l'écran, les empêchant ainsi d'en sortir. Le pourtour de l'écran devient alors une zone particulière où l'utilisateur peu placer les fenêtres dont il n'a pas l'utilité immédiate mais qu'il souhaite conserver à portée de vue, à la manière de Scalable Fabric [8] (Figure 3). Là encore, nous avons veillé à ce que l'opération soit aisément réversible : lorsque l'utilisateur éloigne une fenêtre du bord de l'écran, celle-ci retrouve progressivement son échelle originale.

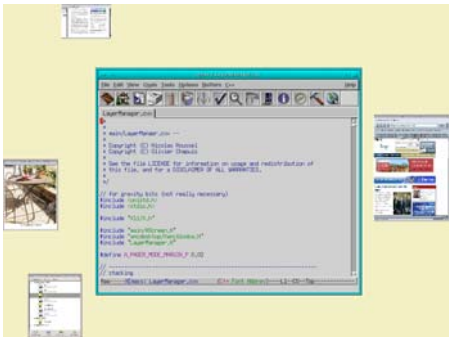

Figure 3 : Réduction progressive d'échelle sur le pourtour de l'écran.

Lorsqu'un déplacement est initié, la fenêtre concernée est automatiquement rendue semi-transparente. Il est ainsi possible de regarder temporairement derrière une fenêtre en cliquant sur sa barre de titre puis en relâchant le bouton sans l'avoir déplacée. L'intérêt de cette opération nous a conduit à implémenter d'autres opérations temporaires et auto-réversibles permettant elles-aussi de voir derrière les fenêtres tout en maintenant leur contenu visible. Un changement d'échelle ou une rotation effectués en appuyant conjointement sur une touche particulière du clavier seront ainsi annulés lors du relâchement du bouton de la souris, ramenant la fenêtre dans sa configuration initiale.

Metisse permet également d'implémenter des opérations s'appliquant à un groupe ou à l'ensemble des fenêtres. Nous avons ainsi implémenté une interface zoomable permettant de placer les fenêtres sur un écran virtuel neuf fois plus grand que l'écran physique. Lorsqu'une fenêtre est poussée en dehors de l'écran physique au-delà d'un certain seuil, Metisse déclenche automatiquement une transition de la vue sur l'écran virtuel. La molette de la souris sert à effectuer un zoom continu, permettant d'obtenir une vue d'ensemble montrant l'intégralité de l'écran virtuel (Figure 4). À la différence du mode *Exposé* d'Apple, l'utilisateur peut alors toujours interagir librement avec les fenêtres. Il peut également déclencher un zoom automatique sur l'une des neuf parties de l'écran virtuel d'un simple clic.

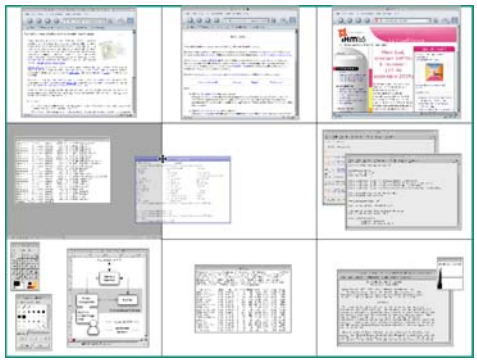

Figure 4 : Vue d'ensemble d'un écran virtuel zoomable.

Une configuration particulière permet d'envisager l'utilisation prochaine de Metisse sur une table interactive [2]. Dans cette configuration, l'écran est séparé en deux zones horizontales. Les fenêtres entièrement situées dans une des deux zones ne subissent aucune transformation. Lorsqu'une fenêtre est déplacée d'une zone à l'autre, une rotation lui est progressivement appliquée jusqu'à atteindre 180 degrés. Metisse permet également de dupliquer une fenêtre, la copie étant ensuite manipulable comme l'original. Cette possibilité permet ainsi à deux utilisateurs se faisant face de regarder ensemble un même document (Figure 5).

Bien que le système ne permette pas encore l'utilisation simultanée de plusieurs périphériques de pointage ou de saisie, la duplication de fenêtre s'avère très utile dans le cas de l'utilisation grandes surfaces d'affichage (e.g. écrans multiples ou écran à très haute résolution). En plus de la duplication telle quelle, Metisse permet la duplication de portions rectangulaires de fenêtres et leur intégration au sein d'une fenêtre nouvelle ou existante (voir [11] pour de plus amples détails).

Figure 5 : Configuration pour une utilisation sur une table interactive. Un utilisateur est en train de déplacer la fenêtre montrant un dessin. Une autre fenêtre (un navigateur Web) a été dupliquée et passée de l'autre côté de la table.

Metisse est actuellement utilisé par la société Mekensleep (http://www.mekensleep.com/) pour la réalisation de Poker3D, un jeu de poker multi-joueurs au graphisme basé sur OpenGL. Le serveur Metisse permet l'intégration d'une application externe de messagerie instantanée au sein du jeu, celui-ci se substituant à la fois à FVWM et à FvwmCompositor. Metisse permet en outre l'intégration d'interfaces traditionnelles 2D dans la scène OpenGL (Figure 6).

Figure 6 : Les éléments d'interface 2D de Poker3D (en bleu) sont gérés par GTK+ et intégrés grâce à Metisse dans la scène OpenGL.

CONCLUSION

Dans cet article, nous avons présenté Metisse, un système de fenêtrage spécifiquement créé pour faciliter la conception, la mise en oeuvre et l'évaluation de nouvelles techniques de gestion de fenêtres. Nous avons décrit l'architecture générale du système et donné quelques précisions concernant son implémentation. Nous avons enfin illustré son potentiel à travers plusieurs exemples d'utilisation.

Metisse est implémenté en C et C++ sur plateformes Linux et OS X. Il est utilisable sur un ordinateur portable de configuration modeste (e.g. un Pentium IV à 2 GHz, 768 Mo de mémoire et une carte ATI Radeon Mobility avec 32 Mo de mémoire). Son code source est disponible sous licence GPL à l'adresse http://insitu.lri.fr/~chapuis/metisse/ et a déjà été téléchargé plusieurs milliers de fois. Nous recevons régulièrement des courriers venant de personnes ayant testé le système. La plupart de ces messages sont enthousiastes et certaines personnes utilisent effectivement Metisse au quotidien. Nous préparons actuellement un questionnaire d'évaluation qui sera distribué avec la prochaine version.

BIBLIOGRAPHIE

1. M. Beaudouin-Lafon. Novel interaction techniques for overlapping windows. *Proceedings of UIST 2001*, pp. 153–154. ACM Press, 2001.
2. P. Dietz and D. Leigh. DiamondTouch: a multi-user touch technology. *Proceedings of UIST 2001*, pp. 219–226. ACM Press, 2001.
3. D. Hutchings and J. Stasko. Revisiting Display Space Management: Understanding Current Practice to Inform Next-generation Design. *Proceedings of GI 2004*, pp. 127–134. Canadian Human-Computer Communications Society, 2004.
4. D. Hutchings and J. Stasko. Shrinking window operations for expanding display space. *Proceedings of AVI '04*, pp. 350–353. ACM Press, 2004.
5. D. Hutchings and J. Stasko. mudibo: Multiple dialog boxes for multiple monitors. *Extended abstracts of CHI 2005, pp. 1471-1474*. ACM Press, 2005.
6. E. Kandogan and B. Shneiderman. Elastic Windows: evaluation of multi-window operations. *Proceedings of CHI 1997*, pp. 250–257. ACM Press, 1997.
7. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998.
8. G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. Hutchings, B. Myers, D. Robbins, and G. Smith. Scalable Fabric: Flexible Task Management. *Proceedings of AVI 2004*, pp. 85–89, 2004.
9. G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The Task Gallery: a 3D window manager. *Proceedings of CHI 2000*, pp. 494–501. ACM Press, 2000.
10. N. Roussel. Ametista: a mini-toolkit for exploring new window management techniques. *Proceedings of CLIHC 2003*, pp. 117–124. ACM Press, 2003.
11. W. Stuerzlinger, O. Chapuis, and N. Roussel. User interface façades : towards fully adaptable user interfaces. Rapport de Recherche 1408, LRI, Université Paris-Sud, France, Avril 2005. 9 pages.
12. D. Tan, B. Meyers, and M. Czerwinski. Wincuts: manipulating arbitrary window regions for more effective use of screen space. *Extended abstracts of CHI 2004*, pp. 1525–1528. ACM Press, 2004.