

École POLYTECHNIQUE
PROMOTION X 2008
ALTIERI Julien

RAPPORT DE STAGE DE RECHERCHE
“FROM A LAPTOP TO A WALL-SIZED DISPLAY”
RAPPORT NON CONFIDENTIEL

Option : Département d'informatique

Champ de l'option : Interaction Homme-Machine

Directeur de l'option : Olivier Bournez

Directeur de stage : Olivier Chapuis

Dates du stage : 18 avril au 18 juillet 2011

Adresse de l'organisme :

Laboratoire de Recherche en Informatique (LRI, INRIA) – *In Situ*

Bat 490

Université Paris-Sud

91400 Orsay

Résumé :

Le succès de l'audiovisuel a littéralement propulsé les technologies de l'image au statut de produit de consommation de masse, plaçant de ce fait la recherche au départ d'une course au meilleur écran, dont les grands tournants furent la couleur, les écrans plats, la HD, et aujourd'hui la 3D. Si la taille brute occupait une place primordiale parmi les critères de succès, la tendance est en train de laisser place à d'autres aspects plus centrés sur l'interaction et la portabilité. Ces systèmes (laptops, netbooks, tablettes, smartphones) sont petits par leur taille, mais tout aussi excellents dans bien d'autres domaines, succès commercial compris. Mais le destin des dispositifs d'affichage géants que nous sommes aujourd'hui capables de construire n'est pas révolu : il faut aujourd'hui reconsidérer leur utilisation. L'objectif de ce stage est d'explorer les symbioses possibles entre les deux écoles.

Summary:

Display technologies skyrocketed the research force to the starting point of a race for the best screen which encountered multiple checkpoints (color, flat displays, HD, and today 3D). While the wideness of a screen used to be one of the most decisive criterion for a good display, today's trend is composed by more interaction-centered and portable issues. These new systems (laptops, netbooks, tabletops and smartphones) are all the more appreciated as they manage to combine powerful features as well as light physical characteristics (weight, size etc.). One current challenge is to forge new utilization scenarios for wall-sized displays. This internship is aimed at exploring the suitable symbiosis for both of the two approaches, apparently paradoxical.

Table des matières

Cadre de travail	p. 04
▪ IHM	p. 04
▪ Serveurs Graphiques	p. 04
▪ Metisse	p. 05
▪ WILD & ZVTM	p. 06
▪ Problème posé	p. 07
Réalisation	p. 08
▪ Un compositeur en ZVTM	p. 08
▪ Meta-compositing	p. 10
▪ Premier scenario : utilisateur unique, navigation dans un « grand desktop »	p. 12
▪ Second scenario : le mur comme espace de travail collaboratif	p. 14
▪ Serveur ZVTM	p. 14
▪ Partage des Fenêtres et vie privée	p. 15
▪ Gestion des fenêtres	p. 15
▪ Interaction sur le mur : « curseur volant »	p. 16
▪ Interaction sur le mur : rebond des événements	p. 16
Réflexions et perspectives	p. 18
▪ Intégration de Metisse dans ZVTM	p. 18
▪ Metisse sur le WILD	p. 18
Références	p. 19

Cadre de travail

IHM

Le domaine de l'interaction homme-machine (IHM ou HCI) crée depuis presque 50 ans de nouvelles méthodes pour utiliser les ordinateurs. Amorcée dans les années 60 par Sutherland au MIT, l'idée d'utiliser la « manipulation directe » fut déterminante car elle conduira, quelques années plus tard, la firme Xerox à inventer le premier « ordinateur personnel » (Star, 1981), dont l'interface innovante (graphique) devait permettre aux non informaticiens de l'utiliser naturellement et intuitivement. Son échec commercial n'enraya pas le processus, puisque Apple sort en 1984 le premier Macintosh, avec lequel le futur géant parvient enfin à lancer le marché. Avec cette réussite fulgurante, l'interaction homme-machine devient un enjeu de taille, clé de la diffusion des ordinateurs au grand public. Elle concerne à ce jour aussi bien le développement des concepts purement liés aux interfaces graphiques (souris, menu, bouton, accélération du pointeur...) que les notions bien plus larges (visualisation de l'information, espaces de travail collaboratif, réseaux sociaux ...), et se positionne de ce fait à la frontière entre l'informatique, la psychologie et le design. Sa principale conférence internationale est CHI et rassemble chaque année 2500 participants.

Serveurs Graphiques

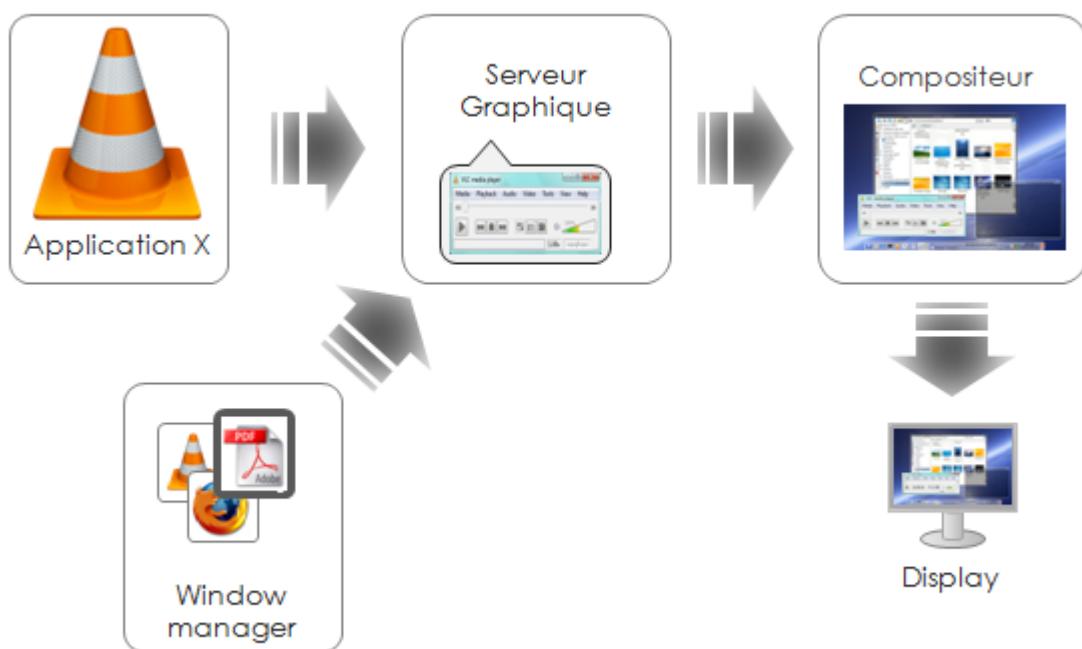
Sous Linux et Mac OS X la composante graphique du système peut être dédiée à un programme tiers appelé serveur graphique. Ce programme communique avec des applications via le réseau (protocole X Window par exemple) afin de les dessiner (fenêtres) et de gérer les entrées clavier et souris (input). Comme ce n'est qu'une couche logicielle de l'architecture, on peut tout-à-fait la désactiver et tout faire en console, ou même la remplacer. Ce découplage facilite en particulier le rendu distant et l'utilisation de multiples supports d'affichage.



Configuration classique

Metisse

Actuellement, les serveurs graphiques sont aussi bien responsables du rendu des fenêtres que de leur affichage effectif dans le bureau, ce qui les rend assez peu flexibles. C'est donc de cette volonté de faciliter le prototypage que Metisse a vu le jour en 2005. Il s'agit d'un serveur graphique « éclaté » déléguant les rôles principaux de l'interface graphique à plusieurs programmes, rendant ainsi l'expérimentation facile à plusieurs niveaux. Le rendu des fenêtres, leur gestion (disposition sur le bureau, interaction avec l'input, ...) et l'affichage final sont dédiés à trois entités différentes : (resp.) le serveur (X-Metisse), le window manager (FVWM) et le compositeur, tous les trois hautement configurables. L'apparition du compositeur dans l'organisation logicielle est particulièrement intéressante car ce dernier peut prendre en charge à lui seul toute la gestion fenêtres sans s'occuper du rendu, puisque le serveur graphique s'en charge. Il devient donc beaucoup plus rapide d'implémenter de nouvelles méthodes d'interaction au niveau « desktop », et ce quelque soit l'environnement (rendu distant).



Configuration du système Metisse

WILD & ZVTM

Les technologies actuelles qui permettent la réalisation d'affichages géants font face à de très grandes difficultés techniques dès que la résolution devient un critère. En effet, aucune carte graphique moderne n'est encore capable de gérer de très grandes résolutions. Le système WILD est une matrice de 32 écrans 30 pouces (8x4), pilotés par 16 machines (2 écrans par machine), ce qui répond au problème. Il s'agit de synchroniser ces machines pour que le système puisse se comporter comme un seul écran virtuel. Ses 21 880 x 7 120 pixels de définition et sa résolution de 100 dpi (contre 40 pour une télévision full HD 40 pouces) lui assurent une qualité d'image exceptionnelle (il n'existe que trois ou quatre murs de ce type dans le monde, alliant à la fois très haute résolution et grande taille). Il est possible de visualiser de très grands jeux de données tels que des images gigapixel aussi bien en détail que dans leur globalité, ce qui laisse place à l'étude de nombreuses problématiques (navigation, visualisation). Le mur est également équipé d'un système de capture haute fréquence (VICON), permettant grâce à des caméras infrarouge, de suivre la position tridimensionnelle d'objets dans la salle du mur avec une précision relativement bonne. Ce système, le WILD Input Server, est au centre de très nombreux travaux actuels.

Il est possible d'écrire des applications Java pour le mur grâce à la librairie ZVTM (Zoomable Virtual Transform Machine), qui gère la synchronisation des écrans et la géométrie de l'ensemble (jBricks), tout en offrant la possibilité de zoomer et de se déplacer dans des espaces virtuels infinis et empilables. Ces fonctionnalités font de ZVTM un outil de première importance pour la visualisation multi-échelle. Une de ses applications en est la navigation dans les images gigapixel telles que la galaxie ou la carte du monde, avec la possibilité superposer des couches d'information (données démographiques, trafic aérien...).



Le WILD : 21880 x 7120 pixels

Problème posé

Le premier objectif du stage est d'exploiter l'opérabilité de ZVTM sur le mur pour y faire fonctionner des applications X-Metisse (applications rendues par le serveur Metisse) et ainsi utiliser le mur comme un dispositif d'affichage pouvant accueillir des fenêtres rendues par un laptop.

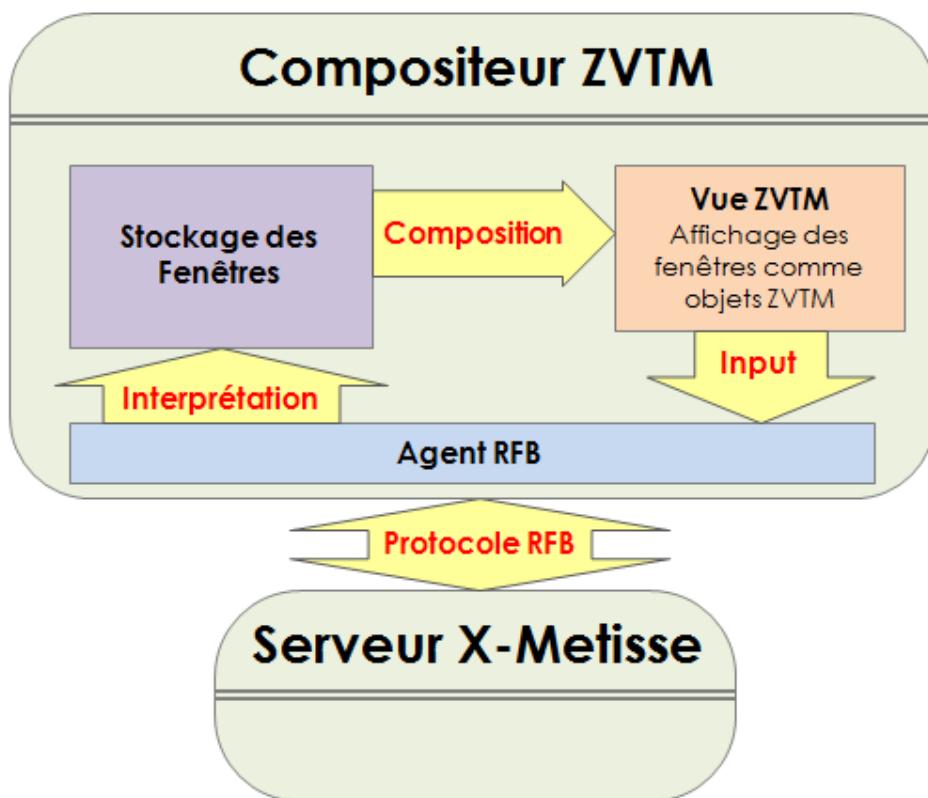
Une fois cette interopérabilité établie, il faut réfléchir à des méthodes d'interaction mettant en œuvre un ou plusieurs laptops et le WILD dans un contexte d'espace de travail augmenté. Il est question d'implémenter et de tester certaines de ces méthodes afin de se rendre compte du potentiel d'une telle association.

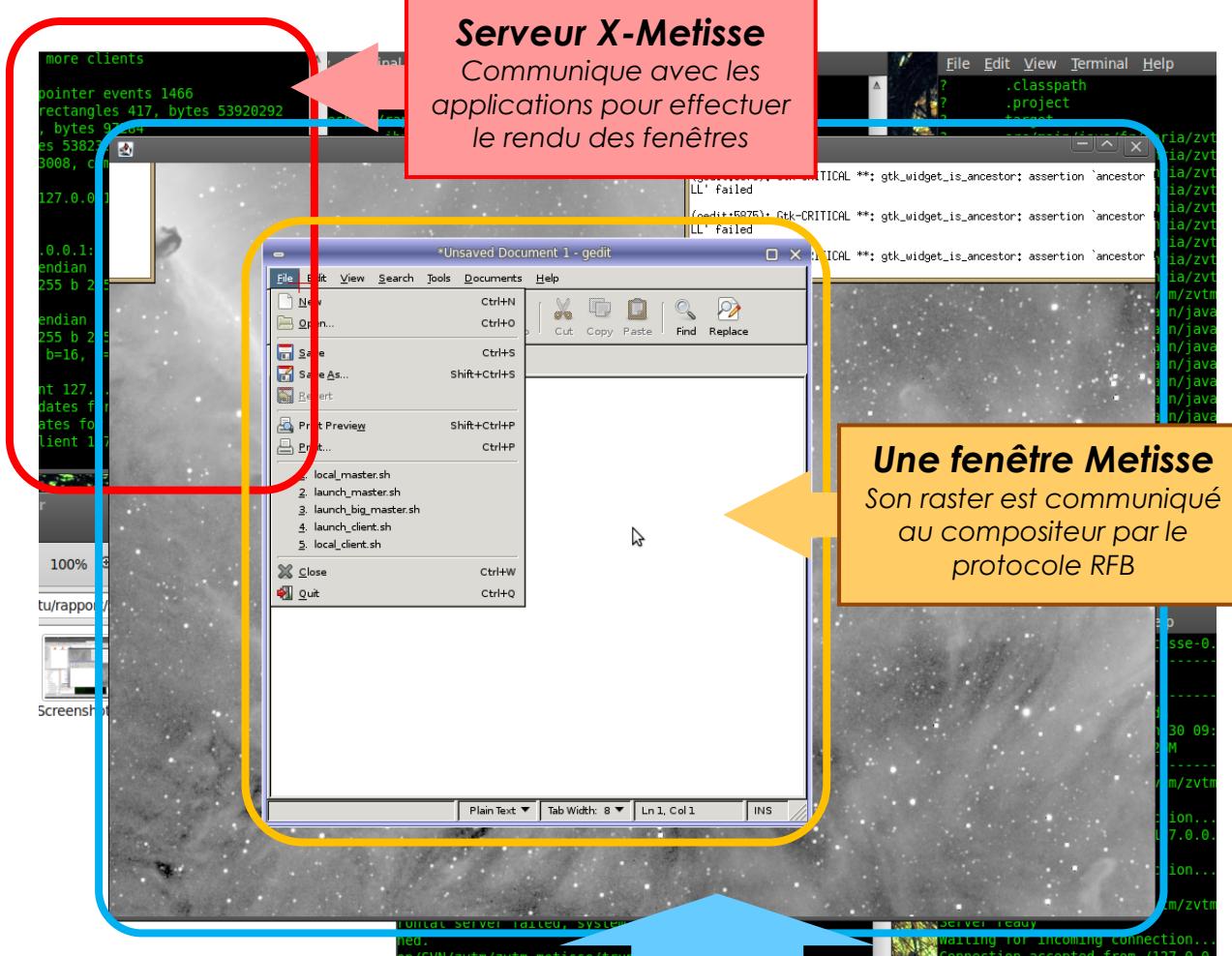
Réalisation

Un compositeur en ZVTM

Le but premier est d'afficher et d'interagir avec des fenêtres X-Metisse dans un environnement ZVTM. La flexibilité de Metisse prend ici tout son sens car il n'y a qu'une seule partie architecturale à réaliser : le compositeur. Lien entre les fenêtres déjà dessinées par le serveur et l'affichage effectif, un compositeur écrit en ZVTM échange avec le serveur Metisse les informations relatives à l'input et aux fenêtres, tout en manageant ces fenêtres et en dessinant le desktop en ZVTM. Lesdites informations sont communiquées par un protocole fondé sur le protocole RFB (Remote Frame Buffer, notamment utilisé par VNC), dimensionné pour une gestion fenêtre par fenêtre. Notons que l'environnement ZVTM permet de naviguer et de zoomer dans ce desktop à l'infini (on ne parle pas encore du déplacement des objets).

Pour interagir avec les fenêtres, il suffit ensuite de rediriger l'input de ZVTM (événements souris, clavier ...) au serveur X, qui transmet lui-même ces événements à l'application concernée. Bien entendu, l'application réagit et envoie de nouvelles instructions au serveur pour se dessiner (highlight d'un bouton etc.). Le déplacement et le redimensionnement des fenêtres étant pris en charge par le serveur X, cette simple combinaison d'affichage et de redirection produit un bureau fonctionnel. Pour le moment, le compositeur s'exécute sur le laptop et fait donc office de « second desktop ».





Un « desktop dans le desktop »

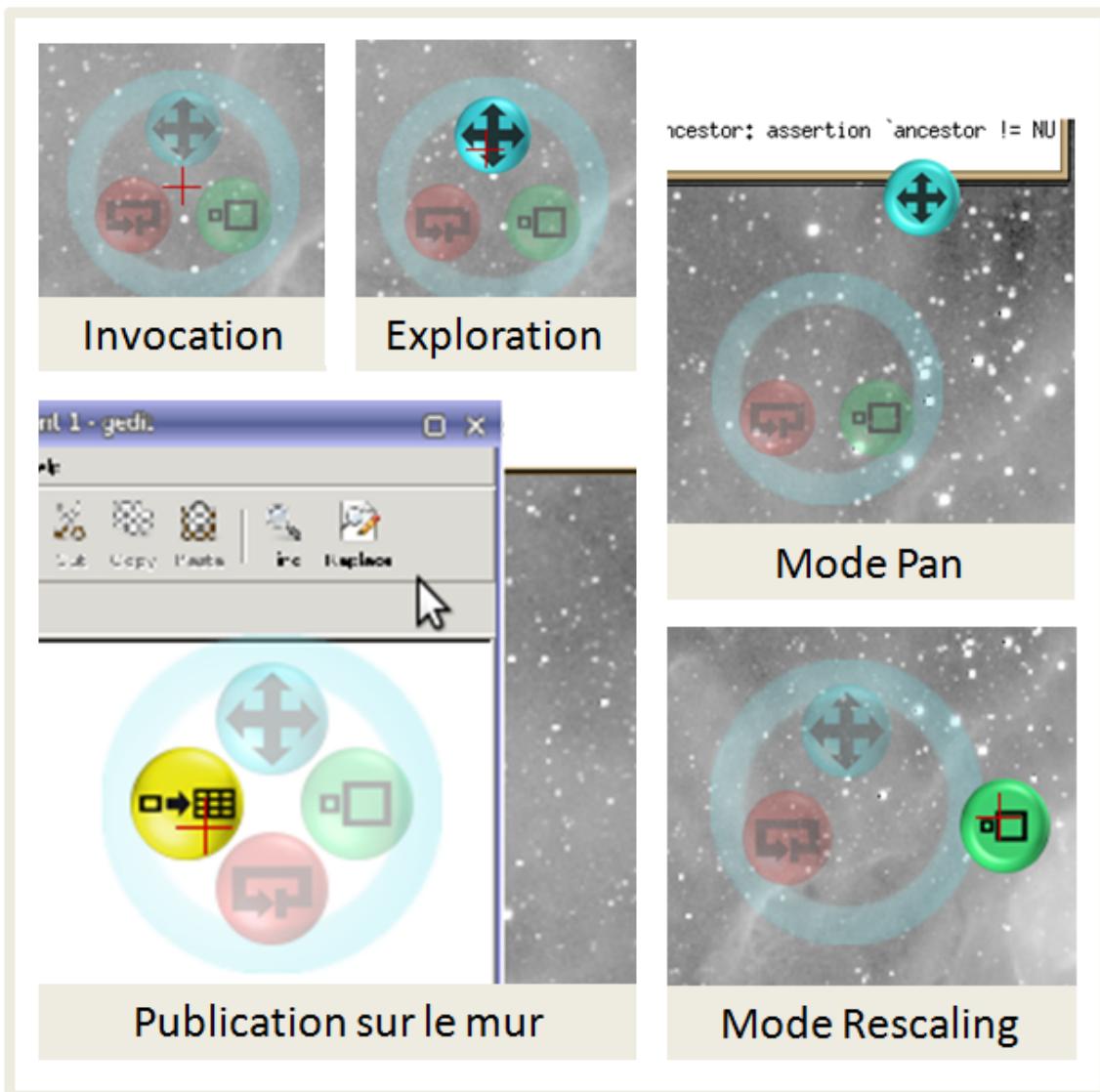
Meta-compositing

À ce stade, l'environnement ZVTM n'est exploité que par soucis de compatibilité avec le mur et ne permet pas de profiter des capacités de navigation propres à ZVTM. Par ailleurs, le seul moyen de déplacer une fenêtre est de la déplacer dans le serveur Metisse, et donc de générer beaucoup de paquets réseaux, alors qu'il serait beaucoup plus performant de modifier la position directement dans le compositeur. Cette difficulté soulève la question du *méta-compositing*, c'est-à-dire comment superposer les interactions du desktop déjà existantes avec les interactions ZVTM.

Une première idée a été d'introduire une touche modale afin d'intercepter les événements intéressants pour la navigation (drag, scroll ...). La combinaison de l'appui sur cette touche avec ces événements permettait ainsi de naviguer dans la vue ZVTM sans interagir avec les fenêtres. Mais cette approche pose un gros problème : la touche que nous choisissons est en fait « sacrifiée » du point de vue du serveur X puisque ses événements ne sont pas redirigés. De plus, le window manager du système d'exploitation intercepte lui-même déjà ce genre d'interaction pour certaines opérations (zoom d'écran, saut d'une application à une autre, bureaux virtuels etc.). Ces actions varient d'une configuration à une autre puisqu'elles sont définies par l'utilisateur, et le seul moyen de nous assurer qu'il n'y ait pas d'interférences avec le système serait « d'ajouter une touche au clavier », ce qui est bien sûr hors de question.

Une seconde possibilité (non implémentée) était de configurer le window manager de X-Metisse (responsable des décorations de fenêtres) pour ajouter des boutons à la barre de titre des fenêtres. D'une part, cela oblige à charger le réseau car le window manager doit communiquer ces nouvelles actions au compositeur ZVTM, d'autre part cette méthode ne s'applique qu'aux fenêtres car le desktop n'a pas de « décorations embarquées ».

La solution pour laquelle nous avons opté est d'utiliser un « PopMenu » au niveau du compositeur ZVTM. Ce menu circulaire apparaît autour du curseur lorsque l'utilisateur le déclenche, et il est possible de l'invoquer aussi bien sur le bureau que sur une fenêtre (les actions qu'il offre en dépendent). Si la question de l'invocation du menu semble aussi problématique que précédemment, la fonction de la touche « sacrifiée » est ici beaucoup plus simple puisque nous avons transformé un « mode » en « événement ». Il suffit donc d'utiliser une touche modale à la façon d'une touche événementielle (comme les lettres) pour assurer un fonctionnement sans interférences. Le menu apparaît (ou disparaît) par un simple « KeyPressed » puis « KeyReleased » sur la touche CONTROL (sans conséquences).

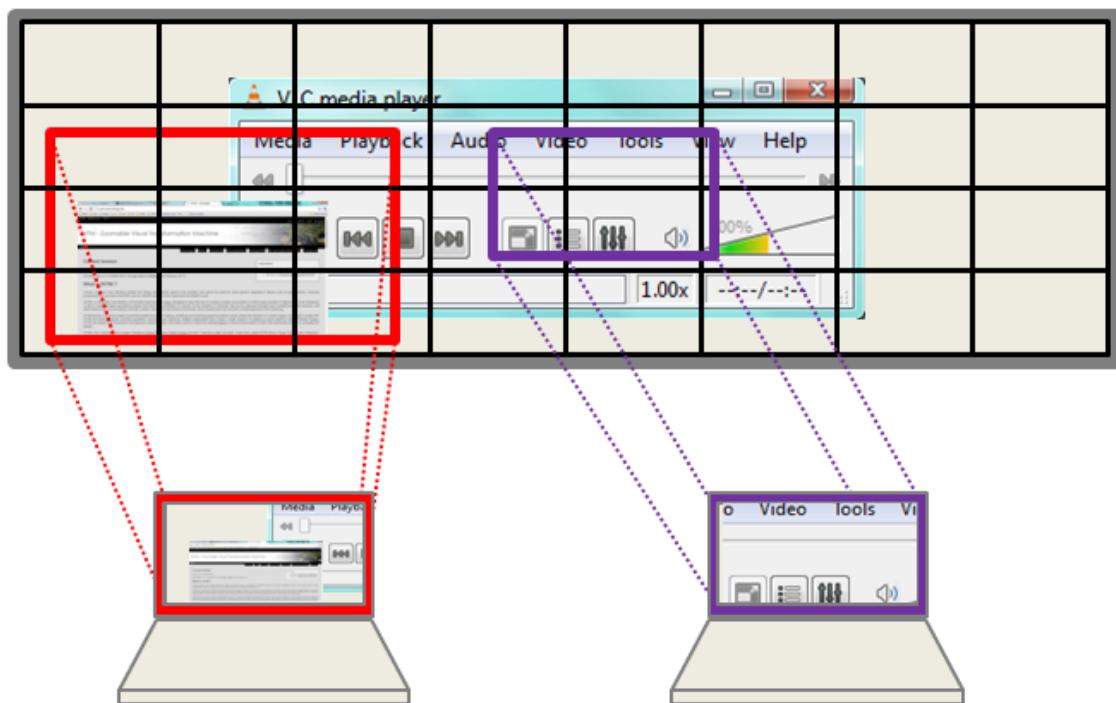


Le PopMenu : pour prendre en charge l'interaction ZVTM

Premier scenario : utilisateur unique, navigation dans un « grand desktop »

L'implémentation déjà en place permet d'afficher la réplique d'une vue ZVTM sur chacune des 16 machines dont la partie visible est décalée et adaptée à la position de chaque machine. L'idée la plus élémentaire était de répliquer la vue du compositeur directement sur le mur, les deux entités partageant ainsi la même « caméra ZVTM ». Une telle configuration pose néanmoins un problème de taille : les fenêtres sont exactement à la même échelle sur le mur (trop petites) et sur le laptop (trop grandes) puisqu'il s'agit de la même caméra.

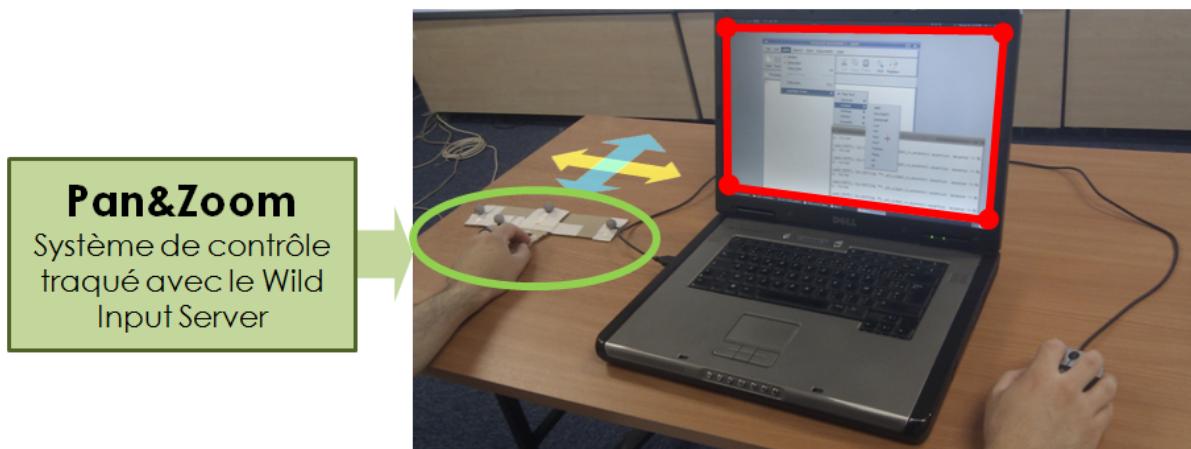
Ce problème conduisant naturellement à vouloir séparer la vue de contrôle de la vue répliquée, nous devions passer à un modèle où les deux entités sont différentes mais partagent un même espace virtuel. Cette configuration permet ainsi d'afficher un espace virtuel (desktop) sur le mur et d'utiliser les laptops comme des télescopes explorant cet espace.



Laptops explorant l'espace virtuel partagé

L'utilisation du WILD Input Server ajoute un *tracking* de la position du laptop pour s'en servir comme un outil physique de navigation, finalisant ainsi la métaphore du télescope (visée avec le laptop d'un point sur le mur). En pratique, si la mobilité latérale est déjà rare (obligation de porter l'ordinateur sur les genoux), les déplacements verticaux sont presque impossibles à réaliser du fait de l'importance de l'angle de vue entre l'utilisateur et le laptop.

Cependant, le concept de contrôle tangible pour la navigation nous a conduits à un mode d'interaction bi-manielle où l'utilisateur utilise une main pour la souris et une main pour se déplacer et zoomer dans l'espace virtuel.



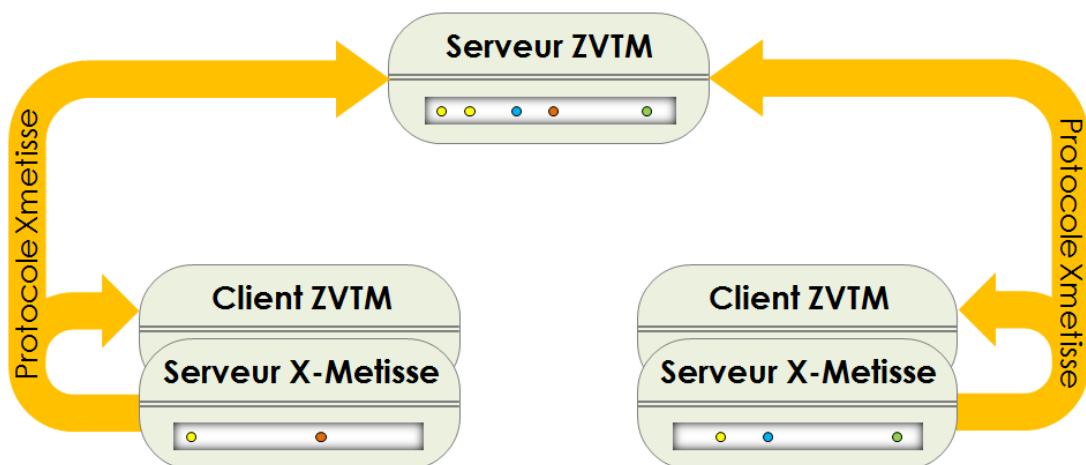
Laptop en mode exploratoire sur l'espace Virtuel. Le déplacement du champ visible est assuré par la main gauche

Second scenario : le mur comme espace de travail collaboratif

Le scenario précédent impliquait de pouvoir séparer le MasterViewer des vues clientes, mais l'espace partagé ne représentait qu'un seul desktop (et donc que ses propres applications). La suite logique de la démarche était donc de passer à un modèle client-serveur complet, où plusieurs utilisateurs peuvent se connecter « au mur » et y partager leurs propres applications.

Serveur ZVTM

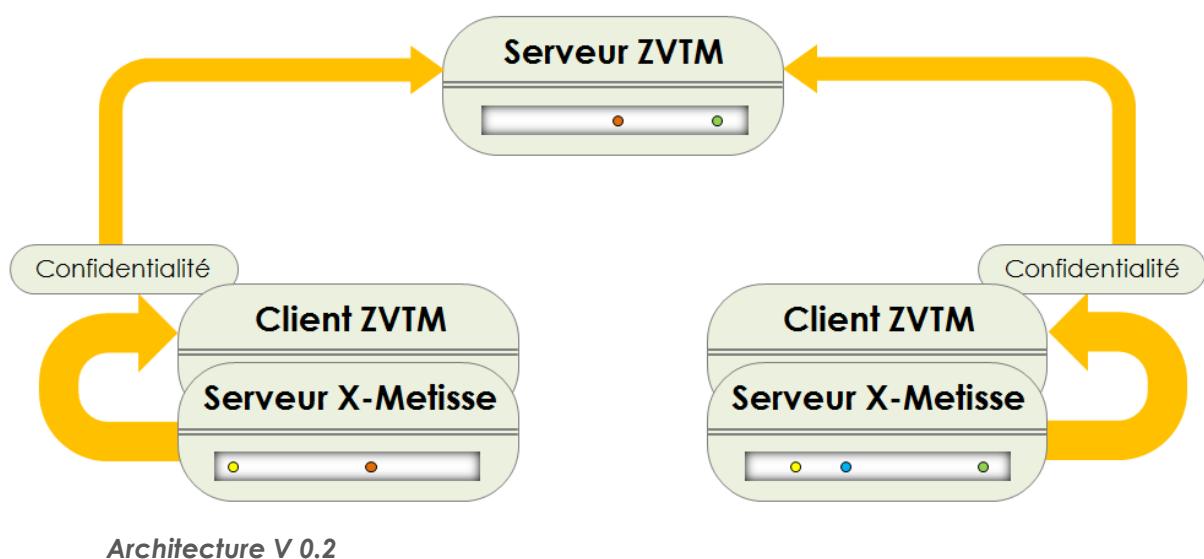
Le MasterViewer fait désormais office à la fois de « serveur ZVTM » et de « client X-Metisse ». Des Clients ZVTM se connectent au MasterViewer et lui transmettent les messages X-Metisse reçus (ajout d'une fenêtre, mise à jour graphique etc.). Le MasterViewer est donc un client X connecté à plusieurs serveurs X en même temps, gérant de ce fait la totalité des fenêtres répliquées sur le mur.



Architecture V 0.1

Partage des fenêtres et vie privée

En réalité, la confidentialité nous a imposé de restreindre la retransmission des messages : seul passent ceux qui concernent les fenêtres que l'utilisateur a choisi. L'utilisateur (client ZVTM) peut donc choisir de « publier une fenêtre », la rendant ainsi visible sur le mur. Cette action s'effectuant grâce au PopMenu que nous avons décrit plus haut, elle est parfaitement orthogonale aux interactions desktop qui changent d'une machine à une autre.

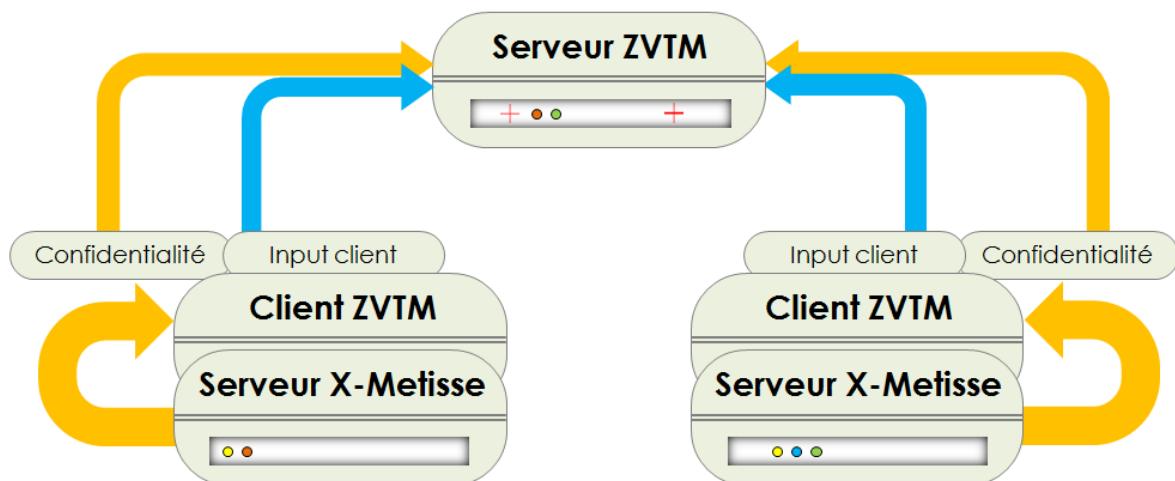


Gestion des fenêtres

Puisque les clients ne font que retransmettre les données du serveur X dont ils dépendent, la modification d'une fenêtre publique sur un laptop (position ou taille) entraîne immédiatement la même modification sur le mur. Pour déplacer une fenêtre de manière indépendante de la plateforme, il faudrait que le mur (ainsi que chaque client) ait sa propre gestion des fenêtres. Mais avec la configuration actuelle, il serait impossible d'agencer les objets sur le mur puisqu'il n'a pas de système d'interaction propre. Pour répondre au problème, nous avons choisi de « libérer » le curseur de l'écran du laptop.

Interaction sur le mur : « curseur volant »

Afin que l'utilisation reste naturelle, nous avons mis en correspondance le bord supérieur de l'écran et le bord inférieur du mur. Le curseur d'un utilisateur peut alors quitter l'écran du laptop par sa partie haute pour se rendre sur l'espace partagé. Dans ce type de situation, l'utilisateur doit pouvoir garder le contrôle sans difficulté, même s'il perd de vue son curseur. C'est pourquoi un second curseur « miroir » apparaît quand le curseur principal a quitté l'écran. Ce curseur est le reflet du véritable curseur (l'axe Y est inversé) par rapport au diopstre. L'avantage du contrôle inversé est qu'il n'y a pas de confusion possible entre les deux modes de fonctionnement. Il suffit donc de porter le curseur miroir en haut de l'écran pour récupérer le vrai curseur.



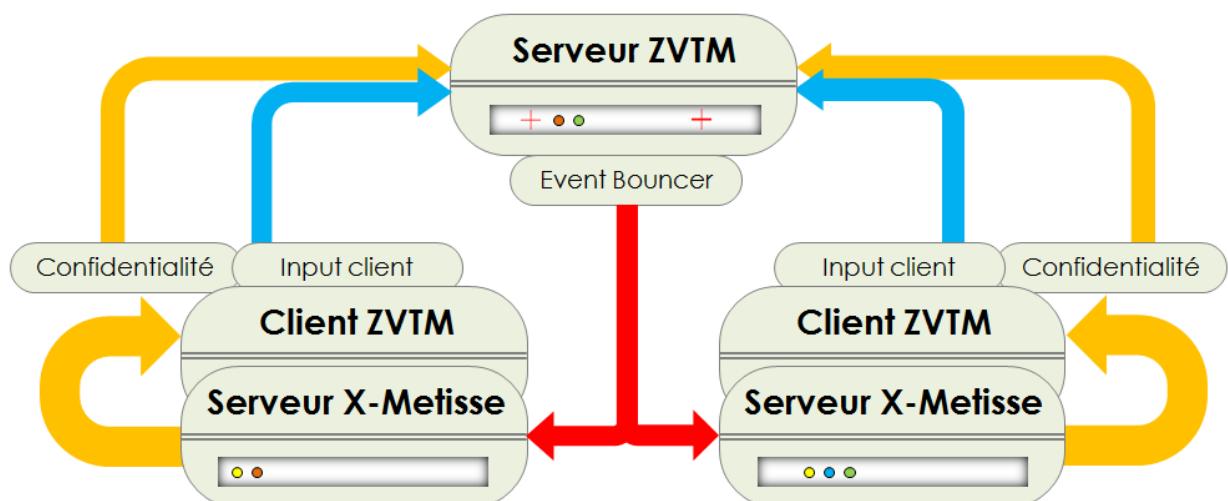
Architecture V 0.3

Interaction sur le mur : rebond des événements

Autoriser l'interaction directe sur le mur nous contraint à de nouvelles modifications architecturales. Désormais, lorsqu'un curseur est sur le mur, tout l'input associé à son propriétaire (clavier compris) doit être envoyé au Serveur ZVTM. Le serveur doit ensuite faire « rebondir » ces événements vers le bon utilisateur.

Prenons un exemple concret : Benoît souhaite accéder à un lien dans une page web qu'Amandine a publié. Il veut le faire sans perturber Amandine, qui est en train de rédiger le compte-rendu de la réunion. Il déplace alors son propre curseur sur le mur pour cliquer sur le lien. Aussitôt, le client de Benoît informe le serveur ZVTM de ce clic, qui connaissant le propriétaire de la page web, fait suivre l'événement au laptop d'Amandine. Le clic de Benoît est alors pris en charge par l'environnement d'Amandine et le navigateur accède à la page demandée.

Les différents utilisateurs peuvent en fin de compte interagir avec toute fenêtre publique, sans interférer avec l'activité de son propriétaire. Ils peuvent également réagencer les fenêtres dans le compositeur mural, indépendamment de la position de ces fenêtres sur les laptops. Nous ajoutons pour finir la possibilité pour le propriétaire d'une fenêtre, de la verrouiller pour qu'il soit le seul à pouvoir interagir avec elle.



Architecture finale V 1.0

Réflexions et perspectives

Intégration de Metisse dans ZVTM

Armée de ZVTM et du WILD Input Server, La visualisation multi-échelle constitue actuellement un domaine des plus actifs dans les applications du WILD. Les applications mettent en scène de véritables groupes de travail tels que des astrophysiciens pour la visualisation des images gigapixel qu'ils produisent. Ce cas précis ne met pourtant en valeur qu'un produit brut, ne permettant que la visualisation et la navigation. L'intégration de Metisse dans ZVTM permettra d'augmenter ce type de scenario, en y superposant des éléments de bureau (page web, éditeurs de textes pour la prise de note etc.), offrant la possibilité pour les utilisateurs de conduire des séances de travail bien plus interactives et collaboratives.

Metisse sur le WILD

Le compositeur ZVTM établit le lien entre cet outil innovant qu'est Metisse et l'espace d'affichage haute résolution géant et interactif que constitue le WILD. Si Metisse fait actuellement du prototypage haute fidélité pour le desktop, son portage sur le WILD élargit son champ d'expérimentation et ouvre les possibilités de recherche sur l'interaction desktop à grande échelle.

Références

Articles :

O. Chapuis and N. Roussel. Metisse is not a 3D desktop! In Proceedings of UIST'05, the 18th ACM Symposium on User Interface Software and Technology, pages 13-22, October 2005. ACM.

E. Pietriga, A Toolkit for Addressing HCI Issues in Visual Language Environments, *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, pages 145-152, September 2005, Dallas, TX, USA

Emmanuel Pietriga, Stéphane Huot, Mathieu Nancel, Romain Primet (2011) Rapid Development of User Interfaces on Cluster-Driven Wall Displays with jBricks Proc. SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2011) ACM

Biehl, I.T., Baker, W.T., Bailey, B.P., Tan, D.S., Inkpen, K.M., Czerwinski, M. (2008). IMPROMPTU: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and Its Field Evaluation for Co-located Software Development. In Proc. CHI 08, 939-948.

Daniel Wigdor, Hao Jiang, Clifton Forlines, Michelle Borkin, and Chia Shen. WeSpace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1237–1246, New York, NY, USA, 2009. ACM.

Jun Rekimoto, "Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments", Proceedings of UIST'97, pp. 31-39, 1997.

Pages des projets :

WILD (Wall-sized Interaction with Large Datasets) :
http://powerfulproxy.com/do_it.php/http://insitu.lri.fr/Projects/WILD

ZVTM (Zoomable Virtual Transform Machine) :

<http://zvtm.sourceforge.net/>