THESE D'HABILITATION

L R I

RAPPORT SCIENTIFIQUE PRESENTE POUR L'OBTENTION D'UNE HABILITATION A DIRIGER DES RECHERCHES

Nicolas ROUSSEL

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

03/2008

Rapport de Recherche N° 1491

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 490
91405 ORSAY Cedex (France)

HABILITATION
À DIRIGER DES RECHERCHES

présentée par

Nicolas Roussel

Discipline : Informatique

Spécialité : Interaction Homme - Machine

Nouvelles formes de communication et nouvelles interactions homme-machine pour enrichir et simplifier le quotidien

7 décembre 2007

Ravin Balakrishnan	University of Toronto	Rapporteur
François Bancilhon	Mandriva	
Saul Greenberg	University of Calgary	Rapporteur
Wendy Mackay	INRIA	
Ian McClelland	Philips Applied Technologies	
Laurence Nigay	Université Joseph Fourier & IUF	Rapporteur
Philips Palanque	Université Paul Sabatier	
Tom Rodden	University of Nottingham	

Habilitation à Diriger des Recherches préparée au sein du
Laboratoire de Recherche en Informatique de l'Université Paris-Sud

Résumé

Mon domaine de recherche est l'Interaction Homme-Machine. Mes travaux dans ce domaine s'organisent autour de deux axes.

Le premier axe concerne la conception de systèmes interactifs pour la coordination, la communication et la collaboration entre individus. Je m'intéresse en particulier à la manière dont des moyens vidéo peuvent être utilisés pour permettre des échanges plus subtils (i.e. légers, nuancés, implicites) et plus informels (i.e. spontanés, opportuns) que ceux permis par les systèmes actuels.

Le deuxième axe concerne la conception de nouvelles métaphores et techniques destinées à enrichir et simplifier l'interaction au quotidien avec les systèmes informatiques. Je m'intéresse plus particulièrement aux moyens de faire évoluer la métaphore du bureau sous-jacente à la gestion des données et des applications dans la plupart des systèmes actuels.

Ce document présente les problématiques liées à ces deux axes de recherche, les travaux s'y rapportant auxquels j'ai participé depuis septembre 2001 et quelques perspectives ouvertes par ces travaux.

Remerciements

Merci à Michel Beaudouin-Lafon, Marie-Claude Gaudel, Wendy Mackay et Claude Puech qui ont su créer au sein de l'équipe *Programmation et génie logiciel* du LRI et du projet *In Situ* de l'INRIA des conditions de travail idéales.

Merci à Ravin Balakrishnan, Saul Greenberg et Laurence Nigay, les rapporteurs de ce mémoire. Merci également aux cinq autres membres du jury : François Bancilhon, Wendy Mackay (encore), Ian McClelland, Philippe Palanque et Tom Rodden. Merci aussi à Yannis Manoussakis, notre délégué aux thèses, pour ses encouragements et conseils avisés.

Merci à Olivier Chapuis et aux nombreuses autres personnes avec lesquelles j'ai eu la chance de collaborer ces dernières années, parmi lesquelles : Caroline Appert, Guillaume Besacier, Stéphane Conversy, Loïc Dachary, Helen Evans, Jean-Daniel Fekete, Nicolas Gaudron, Sofiane Gueddana, Yves Guiard, Heiko Hansen, Catherine Letondal, Emmanuel Nars, Emmanuel Pietriga, Wolfgang Stürzlinger, Aurélien Tabard, Martin Tomitsch, Frédéric Vernier.

Merci enfin à Rosane, Maxime, Thaïs et Sophia, leur présence et leur patience contribuant grandement à l'enrichissement et à la simplification de mon quotidien.

Table des matières

Table des figures

Chapitre

1 Introduction

Mon domaine de recherche est l'Interaction Homme-Machine. Ce domaine couvre à la fois la conception, la mise en oeuvre et l'évaluation de systèmes informatiques interactifs destinés à des utilisateurs humains [Hewett *et al.*, 1992]. Une première partie de mes travaux porte sur la conception de systèmes propres à la coordination, à la communication ou à la collaboration entre individus. Dans ce contexte, je m'intéresse plus particulièrement à l'utilisation d'images fixes ou animées montrant ces individus ainsi que leurs environnements physique et informatique immédiats. Parallèlement à ces travaux liés à une utilisation collective de l'informatique, je m'intéresse également à la conception de nouvelles métaphores et techniques destinées à enrichir et simplifier de manière significative l'interaction entre l'homme et les systèmes informatiques.

Ce mémoire décrit les travaux de recherche que j'ai effectués au sein de l'équipe *Programmation et Génie Logiciel* du Laboratoire de Recherche en Informatique (LRI) depuis mon recrutement comme Maître de Conférences à l'Université Paris-Sud, en septembre 2001, et en tant que membre du projet In Situ de l'INRIA depuis sa création en janvier 2002. Ces travaux s'inscrivant dans la continuité de mes travaux antérieurs, il me semble toutefois utile de revenir brièvement sur quelques éléments mentionnés dans ma thèse [**Roussel, 2000a**][1].

Mes recherches sur les usages de la vidéo pour la communication médiatisée ont été en grande partie inspirées des travaux sur les *mediaspaces*, des systèmes qui assistent un groupe de personnes dans leurs activités quotidiennes en "augmentant" l'espace physique par des moyens audio et vidéo [Bly *et al.*, 1993, Mackay, 1999]. Durant ma thèse, je me suis ainsi intéressé aux problèmes techniques et sociaux posés par ces systèmes, comme la difficulté de déploiement à grande échelle et l'exposition de la vie privée qui en résulte. Ce travail s'est concrétisé par la réalisation de deux nouveaux systèmes, Mediascape et videoServer [**Roussel, 1999**, **Roussel, 2000b**]. Accessibles simplement à travers un protocole basé sur HTTP (Fig. 1.1), ces systèmes permettent de jeter un coup d'œil dans un espace distant ou de partager cet espace pendant plusieurs heures, jours ou mois. VideoServer intègre en outre des mécanismes de notification et de contrôle d'accès per-

[1] Les publications référencées en caractères gras sont celles dont je suis l'auteur ou le coauteur.

mettant de concilier accessibilité permanente et respect de la vie privée. Plusieurs années d'utilisation dans différents contextes, e.g. en France, aux Pays-Bas, au Danemark, en Allemagne et en Suisse, m'ont permis de mesurer l'intérêt de ce type de systèmes pour la coordination et la communication entre collègues, amis ou membres d'une famille séparés par un simple mur, un bâtiment ou plusieurs centaines de kilomètres.

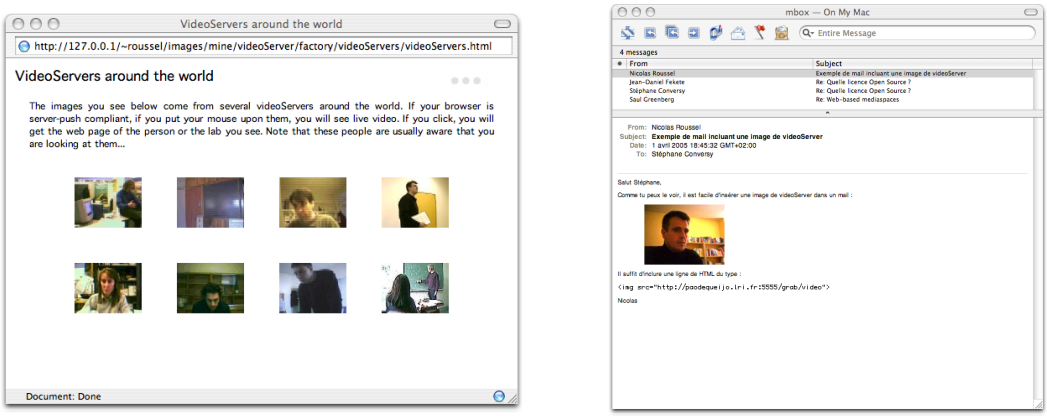


FIG. 1.1: Intégration de services vidéo dans un document HTML affiché dans un navigateur Web et dans un courrier électronique

Mes travaux sur les mediaspaces m'ont amené à concevoir videoSpace [**Roussel, 2001**], une boîte à outils logicielle permettant de faire abstraction des contraintes techniques liées à l'acquisition, au transport et à la restitution des images pour se concentrer sur leurs usages, réduisant ainsi le temps et les efforts nécessaires pour passer d'une idée originale au premier prototype la mettant en œuvre. VideoSpace fut utilisée pendant ma thèse pour réaliser la partie vidéo du *puits* (Fig. 1.2), un système conçu en collaboration avec le service acoustique du CSTB et le projet iMAGIS de l'INRIA pour compléter la visioconférence traditionnelle par des formes de discussion plus informelles [**Roussel, 2002b**, **Roussel, 2002a**]. Composé de trois caméras, trois micros, trois haut-parleurs et d'une surface d'affichage horizontale, le puits permet de discuter de manière conviviale avec un petit groupe de personnes réparties sur plusieurs sites, sa forme originale et un placement ingénieux des éléments techniques limitant les problèmes de cadrage vidéo et de prise de son [**Beaudouin-Lafon *et al.*, 2002b**].

Durant la fin de ma thèse et au cours de mes séjours post-doctoraux, videoSpace fut également utilisée pour expérimenter divers usages de l'image capturée en temps réel du bureau informatique d'un utilisateur. Certains des prototypes réalisés combinaient ainsi l'image de l'utilisateur ou de sa main [**Roussel et Nouvel, 1999**] avec celle de son bureau dans une optique collaborative (Fig. 1.3, images de gauche). Un autre prototype extrayait du flux vidéo du bureau les sous-images correspondant aux différentes applications et les recomposait de manière originale et interactive (Fig. 1.3, image de droite).

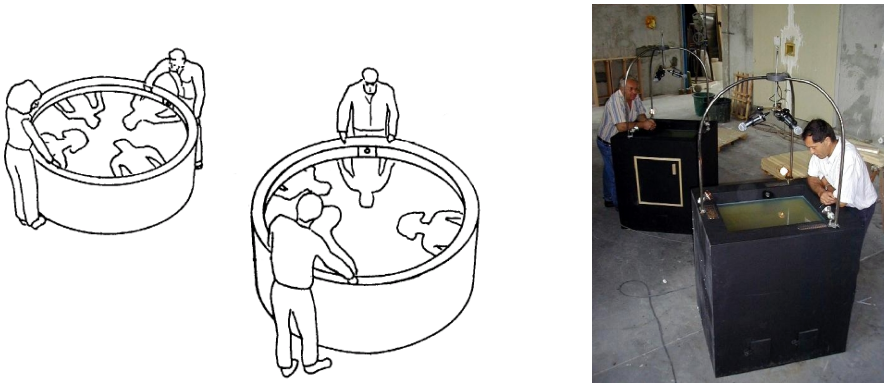


FIG. 1.2: *Le Puits*, concept et prototypes

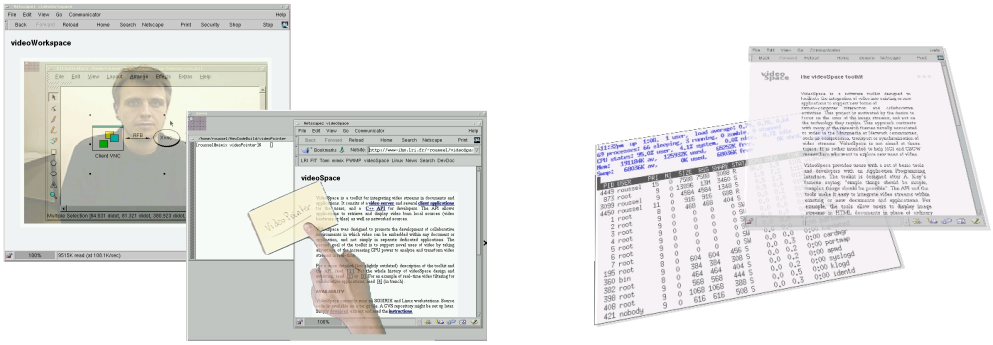


FIG. 1.3: Le bureau informatique comme source d'images

Au printemps 2001, le programme de recherche que je proposais portait sur les thèmes suivants :

- *Traitement d'images pour la communication médiatisée* : inspirée des travaux grenoblois sur les mediaspaces et les interfaces perceptuelles [Coutaz *et al.*, 1998, Bérard, 1999, Crowley *et al.*, 2000], l'idée était d'explorer l'utilisation de techniques de vision par ordinateur pour enrichir les services des systèmes de communication que je développais.
- *L'environnement informatique comme source d'images* : l'objectif était ici de poursuivre mes travaux sur le détournement d'images du bureau informatique dans une optique collaborative, pour pouvoir montrer tout ou partie de son bureau à une personne distante, mais également afin "d'explorer de nouveaux paradigmes de présentation et d'interaction avec les applications et documents informatiques".
- *Nouveaux dispositifs de communication vidéo pour des groupes* : suite au travail réalisé sur le puits, je souhaitais pouvoir à nouveau participer à la conception et à la réalisation de dispositifs de communication de groupe associant logiciel et matériel spécifiques et tirant éventuellement partie des techniques de vision par ordinateur ou des images de l'environnement informatique précitées.
- *Échange et partage d'informations à caractère privé sur Internet* : l'idée était d'essayer de transposer les mécanismes de contrôle et de notification de videoServer à des don-

nées autres que vidéo et, plus généralement, de s'intéresser aux mécanismes pouvant faciliter l'échange et le partage de données à caractère privé sur Internet.

Comme nous allons le voir dans la suite de ce document, les travaux auxquels j'ai participé depuis s'inscrivent relativement bien dans ces thématiques, certaines ayant naturellement été plus explorées que d'autres. Une caractéristique plus intéressante de ces travaux est qu'ils s'inscrivent tous dans l'idée de support (au sens anglais), d'assistance aux utilisateurs dans leurs interactions quotidiennes avec les systèmes informatiques et avec d'autres personnes à travers eux. Le chapitre 2 de ce mémoire est consacré à mes travaux sur la communication médiatisée, tandis que le chapitre 3 porte sur mes travaux liés spécifiquement à l'interaction homme-machine. Ces deux chapitres ont la même structure : ils présentent tout d'abord la problématique de départ et décrivent ensuite les travaux relatifs à cette problématique auxquels j'ai participé. Le chapitre 4 qui conclut ce mémoire présente quelques perspectives ouvertes par l'ensemble de mes travaux.

Les systèmes interactifs n'étant pas toujours faciles à décrire sur le papier, ce document contient de nombreuses figures. Le lecteur est par ailleurs invité à consulter les vidéos illustrant les différents systèmes présentés, accessibles depuis ma page Web[2].

Pour terminer cette introduction, une citation de Bill Buxton résume assez bien l'esprit dans lequel j'ai travaillé au cours de ces dernières années :

> "We have hit the complexity barrier. Using conventional design techniques, we cannot significantly expand the functionality of systems without passing users' threshold of frustration. Rather than adding complexity, technology should be reducing it, and enhancing our ability to function in the emerging world of the future." [Buxton, 1995]

[2] `http://insitu.lri.fr/~roussel/digital-library/metadata/query/?q=roussel`

Chapitre

2 Nouvelles formes de communication

Mon intérêt pour la communication médiatisée a pour origine les travaux réalisés pendant ma thèse sur les usages de la vidéo dans ce contexte. Le premier de ces usages qui vient généralement à l'esprit est la visioconférence...

L'ajout de l'image au son fut envisagé dès les premiers essais de liaisons téléphoniques, à la fin du 19ème siècle, mais il fallut attendre 1964 pour que soit lancé commercialement le premier visiophone, le Picturephone d'AT&T (Figure 2.1). Quarante ans après, les systèmes de communication vidéo sont toujours conçus comme des téléphones améliorés et présentés comme l'outil de communication idéal. Mais contrairement aux prédictions de nombreux futurologues, ces systèmes n'ont pas remplacé les réunions de famille, entre amis ou les voyages d'affaires, le nombre de ces derniers ayant même toujours tendance à augmenter [Jouppi, 2002].

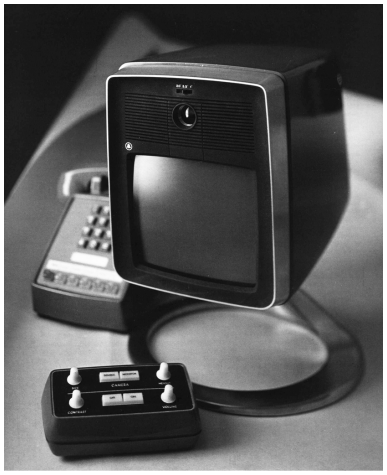


Fig. 2.1: Picturephone [Bell, 1969]

Bien que des logiciels de visioconférence soient aujourd'hui disponibles gratuitement pour les principaux systèmes d'exploitation du marché, leur usage reste très limité. L'échange de texte sous différentes formes (e.g. courrier électronique, messagerie instantanée, mini-messages) et la communication parlée (e.g. téléphonie fixe, mobile ou voix sur IP) restent les moyens les plus utilisés pour communiquer à distance ou de manière asynchrone, pour des raisons personnelles ou professionnelles. Nous assistons ainsi depuis de nombreuses années à l'échec relatif d'une technologie présentée comme le meilleur moyen "d'être cognitivement là-bas sans physiquement y être" alors que dans

le même temps, nous voyons de plus en plus de personnes “être physiquement ici sans cognitivement y être” du fait de leur difficulté à gérer les multiples sollicitations qu’elles reçoivent par l’intermédiaire des autres technologies de communication.

2.1 Problématique

Historiens et chercheurs en Interaction Homme-Machine se sont intéressés aux raisons de l’échec de la visioconférence [Egido, 1988, Lipartito, 2003] et ont proposé des usages alternatifs de la vidéo pour la communication médiatisée. Ces travaux ont en quelque sorte culminé en 1997 avec la publication du livre *Video-mediated communication* [Finn *et al.*, 1997], la plupart des systèmes développés à cette époque étant basés sur des liaisons audio et vidéo analogiques. Mais alors même que la vidéo numérique et les réseaux rapides envahissaient notre quotidien, la fin des années 1990 marqua le déclin de l’intérêt de la communauté IHM pour la communication vidéo. Comme l’avait prophétisé Karam, l’avènement des autoroutes de l’information fut fatal aux divers projets existants [Riesenbach *et al.*, 1995] : les réseaux analogiques furent peu à peu abandonnés et les travaux associés, faute d’être adaptés aux nouveaux réseaux, furent oubliés ou glorifiés en l’état.

Les progrès réalisés depuis dans les domaines des réseaux et du multimedia sont considérables. La bande passante offerte par la technologie ADSL dépasse ainsi celle utilisée au début des années 70 pour transmettre les signaux du Picturephone, tandis que le codec H.264 se veut à la fois “ultra-efficace” et “offrant une qualité d’image sans précédent” [Apple, 2005]. Mais malgré ces progrès, la réalisation de systèmes de communication vidéo reste une affaire de spécialistes et leur évolution est de ce fait principalement guidée par une vision technocentrique et souvent assez naïve du point de vue des usages :

> C’est mieux de se voir quand on se parle !
>
> Les appels visio ? C’est si simple de partager encore plus d’émotions !

Ces slogans publicitaires récents renvoient à la foi inébranlable des concepteurs du Picturephone pour qui l’ajout de l’image au son était à la fois inévitable et nécessairement bénéfique [Bell, 1969, Lipartito, 2003]. Bien que ces suppositions se soient révélées pour le moins exagérées, peu de monde s’interroge aujourd’hui sur le but de la recherche et du développement en matière de systèmes de communication. Pourtant, comme le fait remarquer Lipartito, outre les raisons de l’échec du Picturephone, on peut s’interroger sur celles de son invention. Quel est le but fondamental recherché par ce type de technologie ?

2.1.1 *Être là-bas*

Être là-bas est littéralement impossible. Cette expression renvoie au sentiment de *présence*, que Lombard et Ditton [1997] définissent comme "l'illusion perceptuelle de non-médiation". Ce sentiment varie selon les moyens de communication utilisés. Des théories comme celles de la *présence sociale* [Short *et al.*, 1976] ou de la *richesse des médias* [Daft et Lengel, 1984] ont été élaborées pour caractériser ces différents moyens, les comparer et déterminer ceux qui maximisent l'efficacité ou la satisfaction pour une tâche particulière. La plupart des travaux basés sur ces théories considèrent que le sentiment de présence augmente en fonction de la richesse du média utilisé [Dennis et Valacich, 1999]. Ainsi, la capacité de l'image à transmettre des informations non-verbales (e.g. postures, gestes, expressions, contact visuel) est fréquemment citée pour expliquer l'accroissement du sentiment de présence provoqué par l'ajout d'une liaison vidéo à une liaison audio, cette dernière étant elle-même jugée plus riche qu'une communication textuelle.

L'image étant assimilée à un moyen de renforcer le sentiment de présence, *être là-bas* est généralement le but implicite et rarement remis en cause des concepteurs de systèmes de communication vidéo : "atteindre la même richesse d'information que dans l'interaction en face à face" pour pouvoir "interagir avec ceux qui sont éloignés comme nous le faisons avec ceux qui sont proches" [Hollan et Stornetta, 1992]. Dans leur article intitulé *Beyond being there*, Hollan et Stornetta remettent en question ce but, expliquant qu'il laissera toujours la personne distante en position de désavantage. Selon eux, le but fondamental des travaux sur la communication médiatisée, vidéo ou non, devrait être la conception d'outils qui aillent au-delà de la simple imitation des possibilités offertes par la présence physique.

En décembre 2002, on me proposa de prendre la place d'Austin Henderson à une table ronde sur le thème "*The next five years in telepresence*" lors de la conférence ACM Multimedia pour y apporter un point de vue IHM [**Jain *et al.*, 2002**]. Ce fut pour moi la première occasion de mesurer à quel point les interprétations de l'article de Hollan et Stornetta peuvent différer. En simplifiant, on peut dire que ces interprétations se divisent en deux catégories correspondant à deux sens possibles de l'expression *au-delà* : *plus loin* et *autrement*.

2.1.2 Être plus présent

Hollan et Stornetta posent la question suivante : "que se passerait-il si nous développions des outils de communication permettant une plus grande richesse d'information que le face à face ?". Dans la conclusion de leur article, ils ajoutent : "nous devons développer des outils que les gens préféreront utiliser même dans les cas où la proximité physique est possible". Certains, notamment au sein de la communauté Multimedia, voient ici une invitation à poursuivre plus avant les développements technologiques qui ont conduit aux systèmes actuels : si ces systèmes ne sont pas plus utilisés, c'est simplement qu'ils ne sont pas encore assez bons (pas assez rapides, pas assez fidèles, etc.).

L'ajout de technologie est considéré par les tenants de cette approche comme le principal moyen d'augmenter la valeur du système. Le son et l'image ne sont ainsi considérés que comme une base minimale dont il faut améliorer la qualité et les performances, et qu'il faut éventuellement compléter par d'autres modalités [Rowe et Jain, 2005]. Comme l'illustre la figure 2.2, on a ainsi vu se multiplier ces dernières années les travaux sur des systèmes de prises de vue et de son multiples, sur la reconstruction 3D des participants et/ou de leur environnement et sur des rendus visuels et sonores de type immersif [Gross *et al.*, 2003, Tanaka *et al.*, 2004, Jouppi *et al.*, 2004, Baker *et al.*, 2005, Nguyen et Canny, 2005].

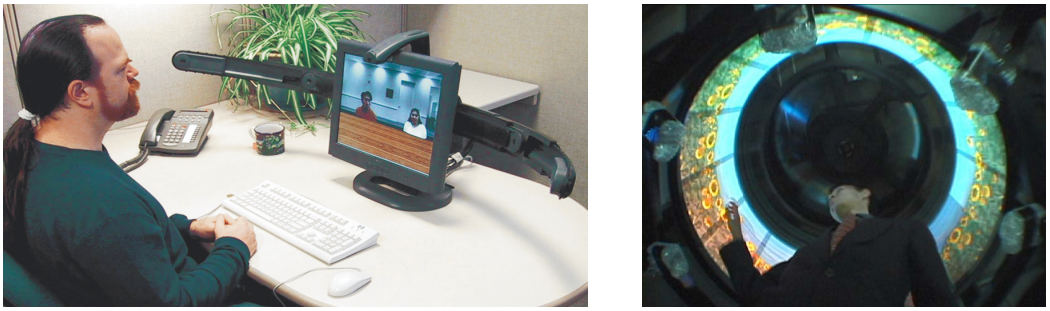


FIG. 2.2: Coliseum [Baker *et al.*, 2005] et Twister [Tanaka *et al.*, 2004]

Mais en essayant de faire "mieux que la présence physique" au lieu de l'imiter, cette approche ne fait que déplacer sans le remettre en cause le but initialement dénoncé par Hollan et Stornetta. Elle place les concepteurs de systèmes dans une situation de course sans fin : les nouveaux moyens mis en œuvre créent souvent de nouveaux problèmes à résoudre, et le simple effet de la loi de Moore permet régulièrement d'annoncer une nouvelle version plus performante et plus riche que la précédente. Enfin, en se focalisant sur les moyens techniques, cette approche néglige les utilisateurs, leurs besoins ou désirs. La complexité des solutions proposées et l'accent mis sur la communication face à face limitent ainsi souvent leur usage à des interactions planifiées, formelles et nécessitant une forte implication des utilisateurs.

2.1.3 Être présent autrement

L'activité d'un groupe ne se limite pas à des situations formelles et à des actes de communication explicites. Diverses études ont montré l'importance d'interactions plus subtiles et plus informelles[1] [Isaacs *et al.*, 1997]. Dans ce contexte, l'analyse de Hollan et Stornetta recommandant de dépasser la présence physique peut être vue comme une incitation à concevoir des systèmes permettant ces autres formes de communication.

[1] L'adjectif *subtil* est ici utilisé pour décrire des formes de communication légères, nuancées, implicites, non intrusives tandis qu'*informel* fait référence à leur caractère spontané et opportun.

L'étude de groupes co-localisés a montré le rôle crucial du canal visuel pour estimer la disponibilité des personnes [Whittaker, 1995]. Le meilleur moyen de savoir si une personne est disponible pour une conversation imprévue n'est-il pas de se déplacer jusqu'à son bureau ? Mais combien de fois va-t-il falloir y aller ? La vidéo permet d'éviter les déplacements inutiles et de voir des personnes hors de portée physique. Différents projets autour des mediaspaces ont ainsi proposé de nouveaux services pouvant avantageusement compléter la visioconférence traditionnelle dans le cadre d'une utilisation quotidienne en milieu professionnel, comme des coups d'oeil de quelques secondes et des liaisons permanentes entre bureaux [Gaver *et al.*, 1992], ou des vues d'ensemble montrant plusieurs bureaux à la fois [Dourish et Bly, 1992]. Ces travaux ont montré que l'utilisation continue mais périphérique de ces services contribue à l'émergence d'une conscience de groupe et renforce le sentiment de présence.

Même périphérique, l'utilisation continue d'un canal visuel ou sonore pose de nombreux problèmes liés à l'exposition de la vie privée. Un certain nombre de solutions ont donc été proposées pour aider les utilisateurs à trouver le compromis adéquat entre accessibilité et intimité, telles que des mécanismes de contrôle et de notification [Gaver *et al.*, 1992], des techniques de filtrage des images et sons transmis [Smith et Hudson, 1995, Zhao et Stasko, 1998, Coutaz *et al.*, 1998] ou des techniques de résumé synthétique de l'activité des personnes [Hudson et Smith, 1996]. D'autres travaux ont également exploré des formes de communication plus abstraites et n'exposant pas l'environnement visuel ou sonore des participants. Strong et Gaver [1996] ont ainsi proposé trois dispositifs minimalistes utilisant la vue, l'odorat et le toucher (une plume, du parfum et un objet vibrant) pour exprimer un sentiment ou une émotion en tirant parti d'un contexte particulier, la relation intime entre deux personnes, pour simplifier à l'extrême la communication. D'autres travaux de recherche sur des dispositifs similaires ont suivi depuis [Brave *et al.*, 1998, Kaye, 2001, Suzuki et Hashimoto, 2004, Brewer *et al.*, 2007], et des produits commerciaux de ce type sont apparus sur le marché (Fig. 2.3).

FIG. 2.3: inTouch [Brave *et al.*, 1998], FeelLight [Suzuki et Hashimoto, 2004] et le lapin communicant Nabaztag commercialisé par la société Violet

Certains chercheurs se sont également intéressés aux récentes évolutions de la communication textuelle, et plus particulièrement à la messagerie instantanée [Nardi *et al.*, 2000,

Isaacs *et al.*, 2002, Grinter *et al.*, 2006]. Au-delà des discussions classiques permises par ce type de système, ces études ont notamment mis en évidence l'intérêt des indicateurs disponibles dans la plupart des outils pour communiquer de manière simple et légère sa disponibilité (e.g. *absent*, *invisible*, *disponible*, *occupé*). Smale et Greenberg [2005] ont par ailleurs montré que les utilisateurs était prêts à détourner certaines fonctions pour pouvoir compléter cette information générique par des indications décrivant de manière plus précise leur activité.

2.1.4 Résumé des questions abordées

Mes travaux récents dans le domaine de la communication médiatisée sont restés fortement liés à l'usage de la vidéo dans ce contexte. Comme on peut s'en douter, ces travaux s'inscrivent dans la démarche générale propre aux mediaspaces, celle visant à communiquer *autrement*, de manière plus subtile et plus informelle que ne le permettent la majorité des systèmes actuels.

Grâce aux outils que j'avais développés [**Roussel, 2000b**], j'avais pu apprécier pendant ma thèse et mes séjours post-doctoraux la vie dans un mediaspace transfrontalier accessible en permanence par Internet. Je pourrais dire "le travail dans un mediaspace", puisque je n'y avais accès qu'au bureau. Mais pour être exact, je devrais dire que comme plusieurs collègues et amis proches, je travaillais dans un mediaspace et qu'en plus de l'usage professionnel que nous en avions, nous en profitions pour maintenir des relations amicales ou intimes. Cette précision peut sembler anecdotique, mais je soupçonne que les mediaspaces à succès ont souvent été construits sur de telles relations, bien que celles-ci soient très rarement mentionnées dans les publications scientifiques.

Depuis septembre 2001, je m'intéresse précisément à la communication entre personnes proches, i.e. qui entretiennent d'étroites relations. Et tandis que la majorité des travaux antérieurs sur la vidéo – y compris les miens – se plaçaient dans un environnement professionnel, je me suis intéressé à la communication en environnement domestique. Ces deux environnements diffèrent sur de nombreux points. L'architecture des lieux et l'usage qui en est fait sont différents. Les personnes qui les habitent le sont également (e.g. leur âge, leur condition physique, leurs motivations, les relations qu'elles entretiennent). En conséquence, parce que majoritairement élaborés dans un cadre professionnel, les paradigmes, méthodes et outils habituellement utilisés pour la conception, la réalisation et l'évaluation de systèmes interactifs se révèlent souvent inadaptés à l'environnement domestique [Hindus, 1999, Crabtree *et al.*, 2002].

Dans ce contexte, les questions auxquelles je me suis intéressé sont les suivantes :

Comment concevoir de nouveaux systèmes de communication ?

> Le cadre domestique rend difficile l'utilisation des méthodes habituelles de conception. L'observation des utilisateurs est par exemple difficile. Dès lors, comment connaître leurs habitudes, la manière dont ils communiquent déjà ? Comment

connaître leurs besoins, leurs désirs ? Comment découvrir les vrais problèmes ou questions auxquelles de nouvelles technologies pourraient répondre ? Les méthodes de conception adaptées à ces problèmes étaient peu nombreuses en 2001.

L'idée que les nouveaux systèmes ne doivent pas viser à reproduire les caractéristiques de la présence physique ne suffit pas à structurer la recherche et le développement. Si elle remet en cause le but implicite sur lequel les travaux antérieurs s'étaient fondés, elle n'en propose pas vraiment de nouveau. Les deux approches qui en résultent ne tendent qu'à complexifier et à diversifier les systèmes. Outre des méthodes de conception appropriées, il manque également un cadre conceptuel qui permettrait de faire le lien entre les nouveaux développements et les outils de communication existants, de les situer les uns par rapport aux autres.

Comment les mettre en œuvre ?

La mise en œuvre de systèmes de communication pour l'environnement domestique pose deux types de problèmes. Tout d'abord, il faut que ces systèmes fonctionnent de manière fiable et robuste. Cela peut sembler évident, mais lorsqu'on est habitué à développer des prototypes dans un laboratoire de recherche, on sous-estime souvent cette première difficulté. Le laboratoire permet de tester les prototypes dans un environnement contrôlé, a priori stable et qui permet une intervention rapide en cas de problème. L'environnement domestique est tout autre.

Ensuite, l'usage domestique impose de fortes contraintes sur le choix des technologies et des techniques d'interaction. L'accès au réseaux téléphonique ou même électrique n'est pas toujours facile. L'encombrement, le bruit, la chaleur et la lumière dégagés sont des facteurs à prendre en compte. La présence d'une caméra ou d'un micro dans une pièce ou à proximité peut poser problème. De simple périphériques d'entrée tels que le clavier et la souris peuvent également être jugés indésirables pour des raisons esthétiques ou pratiques, e.g. par manque de place ou d'appui, rendant difficile l'utilisation des techniques d'interaction habituelles et nécessaire la conception de nouvelles techniques.

Comment les évaluer ?

Les critères d'évaluation habituellement utilisés ont souvent pour origine un cadre professionnel. Ils sont souvent conçus dans une optique de rationalisation de la production, de l'efficacité ou de l'organisation du travail et sont ainsi difficilement applicables dans les environnements domestiques [Crabtree *et al.*, 2002]. Dans ces environnements, les buts et les métriques sont plus difficiles à établir. Il faut donc trouver de nouveaux moyens pour évaluer de manière qualitative, et si possible sur le terrain, les solutions mises en œuvre.

Mes travaux sur ces questions se sont déroulés dans le cadre d'un projet européen puis d'un contrat de recherche avec France Télécom R&D. Deux thèse y ont été associées, l'une sur le point d'être soutenue [Nars, 2007], l'autre toujours en cours. Ces travaux ont fait l'objet de sept publications dont je suis l'auteur ou le coauteur :

– ACM CHI 2003 : [**Hutchinson *et al.*, 2003**]

– IHM 2003 : [**Roussel *et al.*, 2003**] et [**Conversy *et al.*, 2003**] (articles courts)
– Pervasive 2004 : [**Roussel *et al.*, 2004a**]
– IEEE Multimedia, July-September 2004 : [**Roussel *et al.*, 2004b**]
– ACM CSCW 2006 : [**Gueddana et Roussel, 2006**] (*tech note*)
– ACM Multimedia 2007 : [**Roussel et Gueddana, 2007**]

Ces travaux m'ont également donné l'occasion de participer à un atelier de la conférence ACM CSCW [**Roussel, 2006a**] et d'en organiser un dans le cadre de la conférence UbiMob [**Roussel, 2006b**].

La suite de ce chapitre en présente un résumé.

2.2 Sondes technologiques et systèmes de communication pour l'environnement domestique

De septembre 2001 à décembre 2003, j'ai collaboré à interLiving, un projet européen (IST FET, *Disappearing Computer initiative*) qui avait débuté en janvier 2001. Ce projet associait une équipe pluridisciplinaire (sciences humaines et sociales, design et informatique[2]) à six familles, trois françaises et trois suédoises, dans le but de concevoir de nouvelles technologies de communication avec et pour ces familles selon une approche de triangulation [Mackay et Fayard, 1997].

Dès le début du projet, les six familles – une cinquantaine de personnes, enfants, parents et grand-parents – furent engagées dans une série d'activités individuelles et collectives destinées à initier les chercheurs à divers aspects de leur vie quotidienne [Beaudouin-Lafon *et al.*, 2001]. Une série d'interviews, de courtes observations sur site et d'ateliers impliquant une ou plusieurs famille(s) fut organisée. Au cours des ateliers, les familles participèrent à l'écriture de scénarios, à des scéances de brainstorming, à des jeux de conception et à des scéances de prototypage papier/vidéo. Des *sondes culturelles* [Gaver *et al.*, 1999, Gaver *et al.*, 2004] furent également utilisées. Une sonde culturelle est un objet confié à une personne et associé à une tâche à accomplir afin de recueillir des informations de nature inspiratrice sur cette personne et son environnement. Il fut ainsi demandé aux familles d'illustrer les relations et communications entre personnes sur une grande feuille de papier et à l'aide de photos. Ces différentes méthodes furent par la suite appliquées à des publics plus larges, lors de conférences, afin de compléter le travail effectué avec les familles [Mackay, 2004].

A partir des premiers éléments obtenus avec les familles, il fut décidé de centrer les efforts du projet sur la communication entre foyers d'une même famille. Le téléphone et le courrier électronique étaient déjà utilisés pour maintenir le contact et faciliter la

[2]Les partenaires étaient le *Centre for User Oriented IT-Design* (CID) de l'Institut Royal de Technologie de Stockholm, le projet MErLIn de l'INRIA, le groupe IHM du LRI et le *Human-Computer Interaction Lab* (HCIL) de l'Université du Maryland. Pour plus de détails sur ce projet, consulter `http://interliving.kth.se/`

coordination, mais il apparut assez clairement que d'autres formes de communication plus subtiles et plus informelles étaient souhaitées. Afin de mieux comprendre ce besoin, un nouveau type de sonde fut créé : les *sondes technologiques* [**Hutchinson *et al.*, 2003**].

Les sondes technologiques furent conçues pour étudier la manière et les raisons pour lesquelles les membres d'une famille communiquent et pour les inciter à réfléchir, avec les chercheurs, à de nouvelles formes de communication. Basées sur des technologies nouvelles ou existantes, ces sondes doivent être suffisamment simples et flexibles pour servir de catalyseur à l'émergence d'idées et de nouvelles activitées. Contrairement à des prototypes, elles ne sont pas conçues pour être améliorées selon un processus itératif, mais pour être utilisées pendant quelque temps puis abandonnées, les idées et informations recueillies au cours de leur utilisation pouvant ensuite servir de base à la conception et au prototypage de nouveaux systèmes. Les sondes technologiques sont en fait une méthode pour aider les chercheurs à déterminer les technologies qu'il serait intéressant de développer.

Différentes sondes technologiques furent créées et déployées dans les foyers de quatre familles pendant des périodes de plusieurs semaines ou mois. Au cours de ces périodes, des interviews furent organisées et il fut demandé aux membres des familles de noter leurs observations dans des cahiers. Les sondes furent également instrumentées pour conserver un historique des actions effectuées afin de pouvoir reconstituer par la suite leur utilisation dans le temps. L'une des premières sondes développées, MessageProbe, offrait une surface partagée sur laquelle les habitants de différents foyers pouvaient laisser des messages sous forme de Post-It électroniques affichés dans un espace zoomable. Une autre, TokiTok, réagissait aux bruits ambiants en déclenchant sur les autres sondes similaires des lumières et sons.

J'ai pour ma part collaboré à la conception et à la mise en œuvre de deux sondes technologiques (VideoProbe et dans une moindre mesure, TableProbe) et un prototype (MirrorSpace) utilisant la vidéo.

2.2.1 VideoProbe

VideoProbe est un dispositif qui transmet des photos prises de manière automatique dans un foyer à d'autres foyers de la même famille [**Conversy *et al.*, 2003**, **Conversy *et al.*, 2005**]. Il se présente sous la forme d'un boîtier associant une caméra, un écran et deux haut-parleurs reliés à un ordinateur connecté à Internet. Le boîtier peut être posé sur un meuble ou accroché sur un mur, l'ordinateur choisi étant suffisament discret pour passer relativement inaperçu (Fig. 2.4). Le système est conçu de manière à ce qu'on puisse interagir avec lui de la manière la plus simple et directe possible sans nécessiter un contact physique. Des retours visuels et auditifs sont également utilisés pour rendre perceptibles les transitions entre les différents états du système.

Implémenté avec videoSpace [**Roussel, 2001**], VideoProbe "observe" le lieu dans lequel

FIG. 2.4: VideoProbe

il est placé par l'intermédiaire de la caméra. Tant que la scène observée ne change pas, son écran est entièrement blanc (Fig. 2.5). Lorsqu'un changement significatif est détecté, l'écran affiche progressivement les images capturées, à la manière d'un miroir. Si le changement persiste plus de trois secondes, une photo est automatiquement transmise aux autres VideoProbes[3]. Un carré translucide recouvrant progressivement les images indique le temps restant avant la prise de vue. Lorsqu'une photo est transmise, elle est affichée pendant trois secondes correctement orientée et accompagnée d'un son de déclenchement d'appareil photo. Si la scène ne change plus, le système retourne lentement à son état initial. Dans le cas contraire, d'autres photos peuvent être capturées et transmises selon le même processus. Une télécommande permet de basculer le logiciel dans un mode de consultation des photos locales et distantes et de choisir celles que l'on veut conserver. Les autres perdent progressivement leurs couleurs et contrastes et finissent par disparaître au bout de quelques jours (Fig. 2.6).

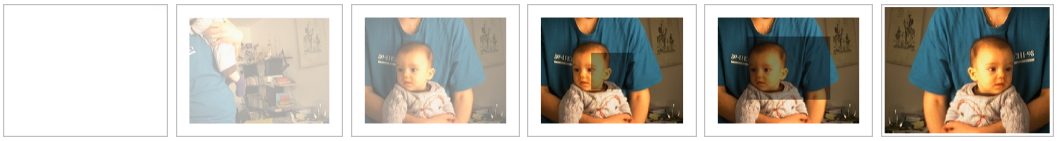


FIG. 2.5: Transitions entre les modes *endormi* (image de gauche), *miroir* et *transmission d'image* (image de droite)

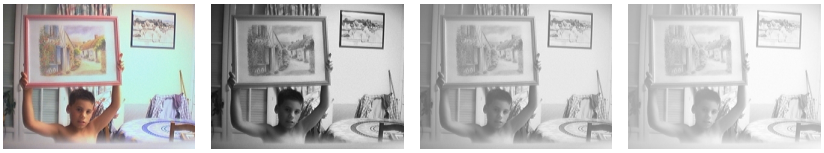


FIG. 2.6: Vieillissement des photos visibles en mode *consultation*

[3]Contrairement à ce que le nom pourrait laisser penser, les VideoProbes n'échangent donc pas flux vidéo en continu, mais de simples photos de temps en temps.

VideoProbe fut dans un premier temps testé dans notre laboratoire et quelques uns de nos domiciles, puis installé dans deux foyers de deux familles françaises pendant environ trois mois. Des cahiers posant un ensemble de questions et permettant de laisser des commentaires libres furent distribués. Les images capturées et un log d'activité furent conservés. Toutes ces données fournirent de nombreux renseignements sur le fonctionnement du système et son utilisation. Les logs d'activité permirent ainsi de repérer les moments où deux VideoProbes d'une même famille étaient utilisés simultanément, au cours d'une conversation téléphonique par exemple. Le déclenchement de la prise de vue par un processus de détection de présence et de mouvement autorise deux formes de communication que l'on distingue aisément dans les images capturées. La première forme, que l'on peut qualifier d'*explicite*, correspond à une utilisation consciente du système afin de transmettre une image particulière : une personne se place devant la caméra et attend volontairement la prise de vue (Fig. 2.7, image de gauche). La seconde forme, *implicite*, correspond à une prise de vue involontaire : une personne entre dans le champ de la caméra sans y prêter attention et reste suffisamment longtemps pour déclencher la prise de vue, ne s'en rendant compte qu'au moment de la notification sonore (Fig. 2.7, image de droite).

FIG. 2.7: Exemples de communications explicite et implicite

Les membres des deux familles exprimèrent spontanément le sentiment de se sentir plus proches des personnes auxquelles elles étaient reliées à travers VideoProbe. En capturant des images qu'elles n'auraient pas eu l'idée ou la possibilité de capturer elles-mêmes, du fait de leur participation à l'action par exemple, VideoProbe fournit un résumé illustré de leur vie quotidienne. L'accès à ce résumé donne le sentiment de partager ce quotidien et suscite rapidement des réactions, le retour pouvant se faire à travers d'autres moyens de communication. Le début de l'utilisation de VideoProbe s'accompagna ainsi d'une augmentation de la fréquence des appels téléphoniques entre foyers. Un procédé de capture purement temporel, à intervalles réguliers, aurait sans doute produit un résumé du quotidien similaire. Mais l'utilisation de la détection de mouvement permet une utilisation plus souple, moins intrusive. Il est ainsi possible de traverser une pièce en étant certain de ne pas être pris en photo : il suffit pour cela de ne jamais rester immobile plus de trois secondes.

VideoProbe s'inscrivit rapidement dans la routine quotidienne, les membres des familles le consultant à la manière d'un répondeur téléphonique pour savoir ce qui s'était passé en leur absence. Il fut utilisé pour laisser des messages explicites, pour indiquer un prochain départ par exemple (Fig. 2.7, image de gauche). On observa également quelques

usages non anticipés, comme l'utilisation explicite de la capture implicite pour prendre des photos au cours d'une soirée organisée pour le nouvel an. Les familles firent quelques suggestions, demandant par exemple s'il était possible d'extraire les photos de cette soirée pour les envoyer à d'autres personnes. Malgré la possibilité d'éviter la prise de vue en restant mobile, la plupart des utilisateurs de VideoProbe demandèrent la possibilité de pouvoir supprimer et empêcher la transmission d'une image embarrassante ou déplaisante. Certains demandèrent également la possibilité de couper temporairement le système, chose qu'ils faisaient de manière détournée en pivotant la caméra vers le mur ou en la masquant avec un objet.

Le déploiement de VideoProbe sur une période relativement longue nous a permis de recueillir des informations importantes pour les trois perspectives qui nous intéressaient dans le projet interLiving (sciences humaines et sociales, design et informatique). Il a fourni des données permettant de mieux comprendre les familles, la manière dont elles vivaient et communiquaient. Il a suscité de nombreux commentaires des familles et des chercheurs qui ont alimenté nos réflexions sur le type de système à concevoir. Enfin, il a posé un certain nombre de problèmes qui nous ont amenés à réfléchir aux infrastructures informatiques nécessaires à ces systèmes. La sous-section 2.2.4 (page 20) reviendra sur ces différentes réflexions.

2.2.2 MirrorSpace

Lors de l'installation de VideoProbe, les familles furent laissées libre de choisir l'endroit où le placer. Comme on pouvait s'y attendre, leur choix se porta sur des pièces communes (salon, salle à manger, cuisine). De manière générale, le choix de la place d'un outil de communication dans une maison est souvent délicat. La mobilité du système peut aider, mais également poser problème : les téléphones sans fil sont rarement là où on en a besoin. VideoProbe ne pouvant pas être facilement déplacé, on sait toujours où il est. Mais les conséquences du choix de son emplacement sont plus importantes que dans le cas d'un téléphone ou d'un ordinateur. Fenêtres et portes font en effet que sa caméra rend potentiellement visibles d'autres parties de la maison sans qu'on en ait immédiatement conscience (Fig. 2.8, image de gauche). On peut ainsi se retrouver dans des situations ou une personne distante est soudain "trop proche" (Fig. 2.8, image du milieu). En même temps, la caméra étant dissociée de l'écran et dans le meilleur cas posée dessus, il est difficile d'être photographié de très près, "les yeux dans les yeux" (Fig. 2.8, image de droite).

Ces deux problèmes, communs à tous les systèmes vidéo, m'ont amené à m'intéresser à la proxémique[4] et à l'utilisation de l'espace autour de ces systèmes. Peu de travaux s'étaient intéressés à cette question auparavant. Ishii et al. avaient remarqué que la distance interpersonnelle perçue à travers leur système ClearBoard (environ un mètre) était

[4]La proxémique – ou proxémie – étudie la manière dont nous utilisons l'espace physique autour de nous dans nos relations avec les autres personnes [Hall, 1966].

FIG. 2.8: Voir plus loin, voir trop loin, ne pas être vu d'assez près

appropriée face à des amis ou collègues, mais pas face à une personne de rang supérieur [Ishii *et al.*, 1993]. Ils avaient donc suggéré la possibilité que le système fournisse un moyen de contrôler la distance perçue. Les auteurs de MAJIC avaient eux remarqué que de nombreux facteurs influent sur cette distance, comme la distance réelle entre la personne et l'écran, la nature du décor, la taille et la qualité des images ou la qualité du son [Okada *et al.*, 1994]. Grayson et Anderson avaient montré que l'utilisation du zoom pouvait changer la distance perçue [Grayson et Anderson, 2002]. Mais personne n'avait vraiment étudié les moyens de contrôler ces différents paramètres.

Suite aux résultats obtenus avec VideoProbe, nous souhaitions développer un prototype permettant la communication vidéo synchrone, offrant quelques garanties concernant la vie privée des personnes et permettant néanmoins des formes de communications intimes. Ce fut pour moi l'occasion de travailler avec les designers du projet interLiving à la conception et à la réalisation d'un système de communication vidéo prenant en compte la notion de distance : MirrorSpace [**Roussel *et al.*, 2003**, **Roussel *et al.*, 2004a**, **Roussel *et al.*, 2004b**].

Comme son nom l'indique, MirrorSpace se présente sous la forme d'un miroir vidéo augmenté (Fig. 2.9). Les flux vidéo des lieux reliés par ce système sont affichés sur un écran unique fusionnant par transparence l'image des participants locaux et distants. Afin de permettre des formes de communication intimes où le regard joue un rôle important, la caméra est placée au centre de cet écran. On peut ainsi s'approcher très près du dispositif tout en restant dans le champ de vision de la caméra et capable de voir les personnes distantes. Chaque dispositif comporte en outre un capteur de proximité qui mesure en continu la distance à la personne ou l'objet le plus proche. Le logiciel, comme dans le cas de VideoProbe, est implémenté avec videoSpace.

Les distances mesurées sur chacun des sites sont utilisées pour appliquer un effet de flou sur les images affichées. Ce flou permet de percevoir l'activité d'une personne éloignée avec un minimum d'implication. Il offre également un moyen intuitif pour initier ou éviter une transition vers un mode de communication plus engagé en se déplaçant simplement vers le dispositif ou au contraire en s'en éloignant (Fig. 2.10, images de gauche). Tandis que les systèmes traditionnels crée un espace partagé correspondant à une distance interpersonnelle particulière, MirrorSpace offre un continuum de distances permettant l'expression d'une grande variété de relations entre individus et permet une

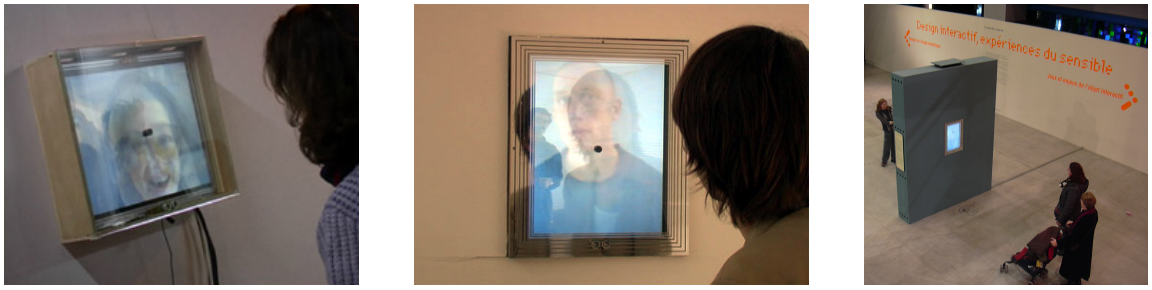


FIG. 2.9: Installations de MirrorSpace à La Villette (*Jeune Création*), à Mains d'œuvres et au Centre Pompidou (*Design interactif - Expériences du sensible*) en 2003

interaction simple et directe pour se positionner, par le mouvement, dans ce continuum.

FIG. 2.10: Réduction de l'effet de flou à mesure de l'approche d'une personne et superposition d'images

Différents prototypes de MirrorSpace ont été présentés au public dans le cadre d'expositions d'art contemporain (Fig. 2.9). Ces expositions nous donnèrent l'occasion d'améliorer progressivement la conception matérielle, logicielle et esthétique du système et d'envisager quelques variations notamment basées sur des modifications du mode de calcul de l'effet de flou à partir des distances mesurées [**Roussel *et al.*, 2006a**]. L'observation de l'interaction entre les visiteurs et le système révéla quelques éléments intéressants. Ainsi, la plupart des personnes ne remarque pas la présence de la caméra au milieu de l'écran, ni celle du capteur de proximité. Elles se retournent par contre lorsqu'elles apperçoivent un autre visage à l'écran, ce qui montre bien que le système est perçu comme un miroir. Les personnes qui se connaissent bien s'approchent très près du dispositif, essaient souvent de se regarder dans les yeux ou de s'embrasser et y arrivent généralement, grâce au placement de la caméra (Fig. 2.10, image de droite). A l'inverse, lorsqu'un inconnu apparaît à l'écran, la plupart des personnes reculent, ce qui rend leur image progressivement floue. Certaines reviennent ensuite lentement vers le système, se dévoilant tout aussi progressivement.

La présentation au public dans le cadre d'expositions n'était pas perçue au départ comme un moyen d'évaluer le système, le contexte étant assez éloigné des environnements domestiques et la configuration permettant dans certains cas de se voir indépendamment

du système. Les observations réalisées me semblent toutefois significatives et plus liées à la nature intrinsèque de MirrorSpace qu'à ce contexte particulier. Elles montrent que le système permet des transitions fluides entre une communication périphérique et une communication intime et qu'une partie au moins du langage corporel peut être utilisé pour contrôler ces transitions. Elles furent en tout cas inspiratrices, étant en grande partie à l'origine de mes réflexions sur le concept de *communication multi-échelles* présenté dans la section 2.4 (page 24).

2.2.3 TableProbe

La sonde technologique TableProbe fut développée dans le prolongement du travail sur la vidéo effectué avec les designers d'interLiving sur VideoProbe et MirrorSpace. Les familles ayant pu tester le premier chez elles et le second lors d'une des expositions, il semblait intéressant de capitaliser sur la sensibilité qu'elles commençaient à développer à ce type de communication. TableProbe est en quelque sorte un hybride de VideoProbe et MirrorSpace, conservant du premier l'idée de permettre une communication par l'image asynchrone et intégrant du second celle de superposition d'images provenant de diverses sources. A l'origine de TableProbe se trouve un logiciel baptisé CVR, pour *Cumulative Video Recorder*. Le principe mis en œuvre par ce logiciel est de permettre la création d'une séquence vidéo par enregistrements successifs, chaque nouvel enregistrement venant se superposer aux images précédentes par transparence ou allonger la durée de la séquence.

La superposition des enregistrements permet de créer simplement des vidéos dans lesquelles une personne apparaît plusieurs fois (Fig. 2.11, image de gauche), ou à une personne de "répondre" à une autre. Une première ébauche de la partie vidéo de CVR fut facilement implémenté à l'aide de la boîte à outils videoSpace. Pour l'interaction, il fut décidé d'utiliser une interface tangible afin de proposer une fois de plus aux familles quelque chose qui ne ressemble pas à un ordinateur. La solution retenue fut de placer des étiquettes RFID dans deux objets permettant de déclencher la lecture et l'enregistrement de séquences vidéo, d'autres objets étiquetés pouvant servir de conteneur pour les séquences (Fig. 2.11, image de droite). La gestion des étiquettes RFID fut elle-aussi rapidement implémentée à l'aide d'une librairie que nous venions de mettre au point[5] et qui était à l'époque l'une des rares, sinon la seule librairie de ce type en Open Source.

La gestion des étiquettes et de leur lien avec les séquences fut ensuite améliorée afin de tirer partie de toutes les possibilités techniques (e.g. lecture simultanée de plusieurs étiquettes) et d'introduire une notion de groupe permettant à plusieurs personnes d'éditer la même séquence. CVR passa également entre les mains des designers qui décidèrent finalement de placer les étiquettes RFID sur des supports plastiques, intégrèrent le lecteur dans une table, l'écran et la caméra dans un coffrage la surplombant et rebaptisèrent le tout TableProbe (Fig. 2.12, image de gauche). Cette version fut utilisée

[5] `http://savannah.nongnu.org/projects/rfid/`

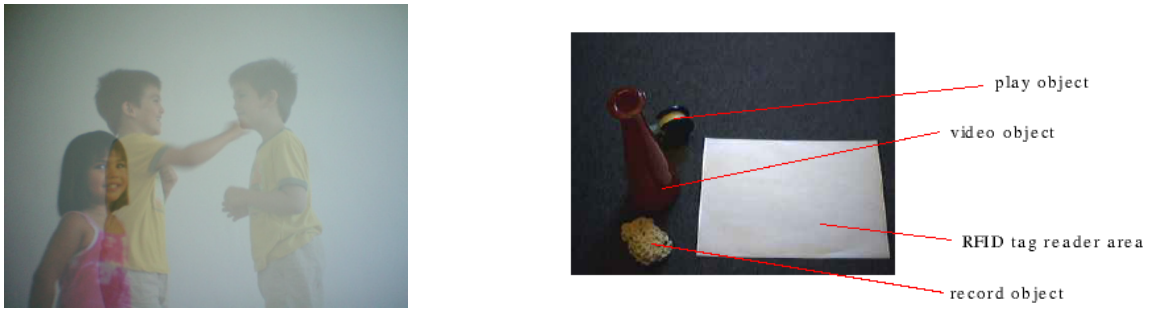


FIG. 2.11: CVR

dans le cadre d'une session participative de la conférence *Tales of the Disappearing Computer* [Mackay *et al.*, 2003] où elle provoqua d'intéressantes réactions [Mackay, 2004].

TableProbe fut ensuite présenté aux familles françaises du projet interLiving dans le cadre d'ateliers participatifs. L'un des enfants présents trouva une ressemblance entre ce système permettant de construire des histoires et le théâtre de marionnettes avec lequel elle jouait. A partir de ses commentaires, de ceux de ses parents et des réactions recueillies lors de la conférence, une version réduite de TableProbe fut créé pour elle : StoryTable (Fig. 2.12, image de droite). Ma propre contribution à l'ensemble de ce travail est assez modeste, puisqu'elle se limite à la partie CVR qui servit de base à TableProbe. Il me semble toutefois que cette histoire méritait d'être contée dans la mesure où elle illustre bien les passerelles et les passages qui existaient au sein d'interLiving entre informaticiens, designers et chercheurs en sciences humaines et sociales.

FIG. 2.12: TableProbe et StoryTable

2.2.4 Enseignements tirés

InterLiving fut incontestablement un succès, atteignant clairement ses objectifs initiaux de travail pluridisciplinaire avec des familles, de développement de méthodes de conception et de création de nouveaux moyens de communication. La démarche associée aux sondes technologiques constitue sans doute l'une des contributions majeures du projet. Selon Google Scholar, l'article [**Hutchinson *et al.*, 2003**] décrivant cette démarche et les

deux première sondes, dont VideoProbe, a déjà fait l'objet de 170 citations. MirrorSpace a lui aussi remporté un certain succès, même si les articles s'y rapportant sont loin de connaître le même sort que celui sur les sondes technologiques. Mais au-delà des résultats diffusés, un important concept a émergé du projet, celui d'*appareil de communication*, accompagné d'un important problème : le manque d'infrastructure logicielle adaptée.

Comme indiqué précédement, l'un des besoins régulièrement exprimés par les familles était celui de disposer de moyens de communication plus subtils et informels que le téléphone et le courrier électronique pour maintenir simplement le contact avec leurs proches. Des couples voulaient un moyen personalisé de garder ce contact, une bague ou une montre par exemple. Des enfants voulaient eux-aussi un contact particulier et direct avec leurs meilleurs amis. Des grands-parents voulaient pouvoir s'adresser directement à leur petits-enfants, en court-circuitant les parents. Le téléphone était généralement jugé trop intrusif pour tous ces besoins et le courrier électronique trop lourd. Les familles souhaitaient quelque chose de plus léger tout en étant prêtes à expérimenter des formes de communication sonores, visuelles ou tactiles qui sortent de l'ordinaire.

Ce besoin régulièrement exprimé nous a amené à définir le concept d'*appareil de communication* (en anglais, *communication appliance*). Comme le résumait l'un des chercheurs suédois du projet, un appareil de ce type doit être vu comme un équivalent du grille-pain pour la communication : un objet simple, à fonction unique. [Nars, 2007] propose une définition un peu plus formelle, adaptée de [**Conversy *et al.*, 2005**] :

> Les appareils de communication sont des systèmes simples à utiliser, comportant peu de fonctions, qui permettent à des personnes de communiquer de manière active ou passive avec un ou plusieurs proches. L'information échangée peut être de nature diverse (sonore, imagée, textuelle, voire haptique ou olfactive). Le style de communication peut varier entre des échanges synchrones de premier plan et la perception périphérique de l'activité des personnes. Cette communication peut se faire à travers l'espace, entre différents foyers par exemple, mais aussi à travers le temps, par l'intermédiaire de "notes" à courte durée de vie ou la préservation implicite des données échangées sur de plus longues durées.

VideoProbe, TableProbe, MirrorSpace et les autres sondes technologiques et prototypes du projet interLiving furent développés dans cet esprit. D'autres chercheurs avaient auparavant réalisés des prototypes dans le même esprit, comme la plume, le diffuseur de parfum et l'objet vibrant de Strong et Gaver [1996] ou les Digital Family Portraits [Mynatt *et al.*, 2001]. Mais très peu de travaux s'étaient intéressés à l'infrastructure nécessaire pour déployer et utiliser ce type d'application.

Le déploiement de VideoProbe fut nettement plus complexe que nous ne l'avions imaginé. L'installation de lignes ADSL prit un certain temps. Il apparut ensuite que l'adresse IP allouée par le fournisseur d'accès n'était pas fixe. Un service public de DNS dynamique fut utilisé pour pouvoir administrer les machines à distance, et il fut décidé d'utiliser un serveur du LRI comme intermédiaire aux échanges de données. Un mécanisme de synchronisation de répertoire fut mis en place à travers ce serveur afin que

chaque VideoProbe puisse fonctionner de manière indépendante en cas de coupure réseau, les images capturées étant ensuite transmises aux autres à la première occasion. D'autres problèmes liés au contexte particulier des installations furent également rencontrés, comme des interférences avec un système d'alarme téléphonique.

Les contraintes de bande passante et de synchronisation étant bien plus fortes dans le cas de MirrorSpace, il fut décidé dès le départ de se restreindre à une implémentation sur réseau local et les transmissions de données furent implémentées à l'aide de technologies multicast. Dans le cas de VideoProbe, les liaisons entre dispositifs étaient décrites dans un fichier de configuration sur le serveur central. Des protocoles de type Zeroconf[6] furent utilisés pour MirrorSpace afin que les instances du système puisent se trouver automatiquement sur le réseau. Mais au-delà des aspects techniques, les usages imaginés par les familles vinrent rapidement compliquer les choses...

Dans l'optique d'une généralisation de l'usage des appareils de communication, les familles exprimèrent rapidement le souhait de pouvoir contrôler finement les personnes avec lesquelles elles communiquent. La notion de *groupe* s'imposa comme un moyen simple d'organiser ces communications. Les groupes envisagés étaient de petite taille et constitués de personnes proches en fonction d'intérêts communs ou d'attirances personnelles, semblables aux *réseaux sociaux intimes* évoqués par Aronson [1971]. En raison de ce caractère intime, ces groupes devaient être privés (sans spam) et sûrs (sans espion). Il fallait aussi des interfaces simples, pour que chaque membre de la famille puisse créer et administrer ses propres groupes et communiquer avec eux. Mais les technologies susceptibles de répondre à ces besoins, les réseaux privés virtuels par exemple, étaient beaucoup trop compliquées pour des personnes ordinaires. Un nom fut choisi pour désigner l'ensemble de cette infrastructure qui faisait cruellement défaut : FamilyNet. Une interface tangible à base de cartes et de technologie RFID fut proposée pour représenter les groupes et les manipuler [**Beaudouin-Lafon *et al.*, 2002a**]. Un premier prototype fut réalisé [Nars, 2003] puis un second, plus sophistiqué, alors que le projet se terminait [**Sundblad *et al.*, 2004**, Mackay *et al.*, 2004, Mackay et Beaudouin-Lafon, 2005]. FamilyNet servit également de point de départ à une thèse qui sera évoquée dans la section suivante [Nars, 2007].

2.3 Outils et interfaces pour la communication de groupe

Contrairement à ce que l'on pourrait croire, il est aujourd'hui plus difficile de développer et déployer des applications de communication qu'il y a 11 ans, lorsque je débutais ma thèse, ou même 6 ans, lors du démarrage d'interLiving. La bande passante disponible a certes été grandement multipliée. Les technologie multimedia ont également progressé. Mais l'environnement réseau est beaucoup plus complexe qu'auparavant. Fin 1997, je pouvais aisément transmettre des flux vidéo entre n'importe quelles machines du LRI

[6] `http://www.zeroconf.org/`

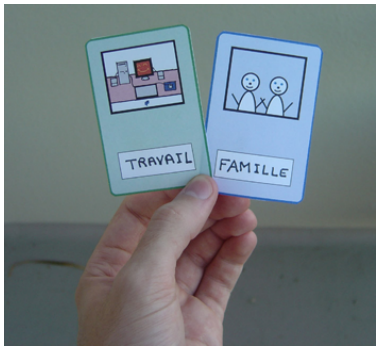


FIG. 2.13: Exemples de cartes utilisées par [Nars, 2003]

et de l'institut RIACA de l'Université d'Eindhoven, aux Pays Bas. Mais les mécanismes de protection et les routeurs faisant de la traduction d'adresse (NAT) se sont depuis multipliés, rendant les connexions pair à pair beaucoup plus difficiles. On se trouve ainsi dans une situation paradoxale où deux personnes disposant chacune d'un modem "triple play"[7] et d'une caméra peuvent très bien suivre le même programme télévisé à travers le modem et être en même temps incapables d'échanger la moindre image de leur caméra.

Le succès d'Internet, le retard de mise en service d'IPv6 et la sécurisation des systèmes ont eu pour fâcheuse conséquence une modification des rapports entre machines. D'un Internet de pairs dans lequel chaque machine pouvait être jointe par toutes les autres, nous sommes passés à un réseau de services faisant une distinction entre fournisseurs et clients. Cet Internet est structuré de manière à ce que les clients puissent accéder facilement aux services des fournisseurs, non pour qu'ils le deviennent eux-mêmes. Malheureusement, les environnements domestiques qui m'intéressent sont du côté client. Et à cette difficulté s'ajoute le fait que le réseau domestique est lui même de plus en plus complexe [Grinter *et al.*, 2005]. L'environnement réseau est donc sauvage des deux côtés du modem...

John Walker, fondateur d'Autodesk et créateur du logiciel de téléphonie *Speak Freely* décida en janvier 2004 d'en abandonner le développement précisément pour ces raisons[8]. Je n'ai pour ma part toujours pas jeté l'éponge. Au fil des années, la boîte à outils videoSpace a été enrichie de nombreuses fonctionnalités sans rapport avec la vidéo dont plusieurs relatives au réseau, au point que le nom videoSpace fut abandonné à la fin du projet interLiving. Cette boîte à outils est aujourd'hui diffusée sous licence LGPL sous le nom Núcleo[9]. Permettant dès l'origine des échanges par UDP, TCP et HTTP, elle implémente en outre aujourd'hui les protocoles DNS-SD (découverte de services sur réseau local), STUN (découverte de l'adresse publique d'un routeur NAT) et XMPP (présence et messagerie instantanée). Núcleo fut notamment utilisée par Emmanuel Nars au cours de sa thèse, débutée en septembre 2003 [Nars, 2007].

[7] Un modem ADSL fournissant l'accès à Internet et à des services de téléphonie et télévision
[8] `http://www.fourmilab.ch/netfone/windows/speak_freely.html`
[9] `http://insitu.lri.fr/~roussel/projects/nucleo/`

Le point de départ de cette thèse est le besoin identifié sous le nom de FamilyNet dans le cadre d'interLiving : celui d'une infrastructure adaptée à l'utilisation des appareils de communication à travers la notion de groupe. E. Nars s'est attaché à préciser cette notion, en s'intéressant à la manière dont on communique aujourd'hui avec plusieurs personnes et en proposant une liste de propriétés à satisfaire. Cette analyse montre que les solutions techniques disponibles sont insuffisantes, se résumant généralement à l'utilisation de listes d'adresses difficiles à maintenir et à synchroniser entre dispositifs et entre membres du groupe. Partant de ce constat, la thèse propose de rendre la notion de groupe accessible comme objet de première classe directement manipulable par le programmeur et les utilisateurs à travers une nouvelle infrastructure baptisée Circa.

Plusieurs versions de Circa furent réalisés au cours de la thèse [Nars, 2004a, Nars, 2004b, Nars, 2005]. La partie réseau de la version actuelle est basée sur le protocole XMPP [Saint-Andre, 2004a, Saint-Andre, 2004b]. L'implémentation des groupes repose sur plusieurs extensions de ce protocole permettant la création de salons de discussion, le stockage de messages reçus en absence et la publication d'informations structurées. La librairie RakNet [10] est utilisée pour les échanges de données binaires entre applications membre d'un même groupe, les données textuelles pouvant être transmises par le canal XMPP.

Circa libère le programmeur des contraintes liées à la transmission de données en permettant à chaque application de communiquer avec un nombre arbitraire de groupes et à chaque groupe d'être utilisé sur un nombre arbitraire d'applications. Quelques lignes de code suffisent pour créer un groupe, y ajouter des membres et l'utiliser. Une interface à base de cartes et d'étiquettes RFID dérivée de [Nars, 2003] offre des fonctionnalités identiques aux utilisateurs. Circa a été utilisée par E. Nars pour prototyper quelques applications de démonstration ainsi que par d'autres membres de l'équipe In Situ [Wauthier, 2006, Riche et Mackay, 2007]. Ses fonctionnalités seront probablement intégrées à Núcleo dans le futur. Elles ouvrent en tout cas de nouvelles pistes de réflexion concernant, par exemple, la conception d'interfaces adaptées à des communication multi-groupes, et non plus seulement multi-utilisateurs.

2.4 Vers une communication multi-échelles

Les réflexions et observations faites à travers MirrorSpace sur la gestion de l'espace dans les situations de communication vidéo m'ont amené à m'intéresser à la notion d'engagement dans ces situations. Le concept de *degré d'engagement* est plusieurs fois mentionné dans les travaux sur les mediaspaces. Gaver et al. [1992] en proposent une définition assez simple : "le degré d'intérêt commun[11]". D'autres chercheurs ont raffiné ce concept ou proposé des concepts similaires [Fish *et al.*, 1993, Greenhalgh et Benford, 1995]. La définition que je propose, inspirée de ces variantes et de la proxémique, est la suivante : "la

[10] `http://www.rakkarsoft.com/`

[11] *the extent to which a shared focus is involved*

limite dans laquelle une personne est prête à s'exposer et à être sollicitée par d'autres" (Fig. 2.14).

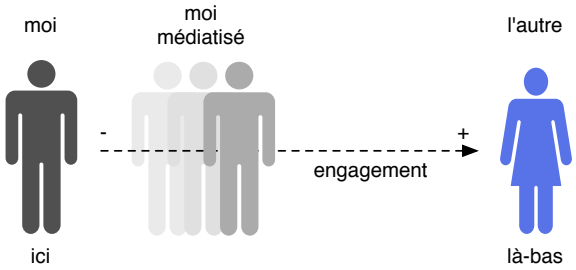


FIG. 2.14: Le concept de degré d'engagement vu comme un potentiel d'exposition et de sollicitation

Plusieurs degrés d'engagement peuvent être distingués. Le degré permis par un service de communication est lié au niveau de détail offert par celui-ci : plus ce niveau est élevé, plus je suis susceptible d'être exposé et sollicité. Le degré effectif du service dépend bien entendu de la fréquence des échanges et de leur nature exacte (e.g. une caméra dont on a masqué l'objectif n'expose plus les personnes alentours). Le degré recherché par une personne est variable, dépendant du contexte, et peut donc changer en cours de communication :

> Imaginons que je vienne de faire des modifications à un article et que je veuille en informer mon coauteur, dans une autre ville. Ne le voyant pas disponible dans la messagerie instantanée, je commence à lui écrire un courrier électronique. Alors que je suis en train d'écrire, il apparaît soudain disponible. Je change donc d'outil et le contacte par la messagerie instantanée. Je lui explique que j'ai fait quelques changements. Il me répond qu'il a un peu de temps pour en discuter, nous convenons alors de passer au téléphone. Je l'appelle. Quelques minutes plus tard, toujours au téléphone, nous éditons ensemble l'article.

Comme le montre cet exemple, le degré d'engagement consenti par les participants influe sur le choix du service de communication[12]. Celui-ci résulte donc souvent d'une négociation. Il faut également noter que chaque changement de service peut entraîner un surcroît de travail articulatoire, au sens de [Schmidt et Bannon, 1992], en nécessitant par exemple l'établissement de nouvelles connexions. Ainsi, pour passer de la messagerie instantanée au téléphone, il me faudra sans doute composer le numéro de mon coauteur et donc le connaître, le (re)trouver ou le lui demander. Et pour travailler ensemble sur l'article, nous utiliserons peut-être un éditeur partagé qui aura lui aussi sa propre procédure de connexion.

L'association de différents services au sein d'un même système peut dans certains cas limiter ce travail d'articulation. Certains outils initialement conçus pour la téléphonie

[12] D'autres facteurs peuvent entrer en ligne de compte. Dans certains cas, les outils de messagerie instantanée sont ainsi préférés au téléphone en raison de l'archivage automatique des discussions qu'ils proposent.

sur IP peuvent ainsi être utilisés pour de la messagerie instantanée ou de la visioconférence. Mais le simple fait d'associer des services ne garantit pas un passage aisé entre ceux-ci. Les téléphones portables, par exemple, combinent des services d'identification de l'appelant, d'échange de messages textuels, de messagerie vocale, de téléphonie et éventuellement de visiophonie. Cet éventail de services correspond à des degrés divers d'engagement qui permettent de contrebalancer l'accessibilité permanente liée à la possession du terminal. Mais les transitions entre ces services sont généralement difficiles, voire impossibles. Lorsqu'on vient de manquer un appel, par exemple, l'expérience montre qu'il vaut mieux attendre avant de rappeler le correspondant si l'on ne veut pas que sa messagerie décroche pendant qu'il parle à la nôtre.

Bien que les raisons des transitions entre moyens de communication soient régulièrement discutées [Nardi *et al.*, 2000, Isaacs *et al.*, 2002], peu de travaux se sont intéressés aux moyens de les faciliter. La fluidité de ces transitions est pourtant essentielle pour pouvoir alterner rapidement entre des activités de coordination, de communication et de collaboration. Les travaux sur la vidéo ont montré que sous ses différentes formes, elle peut servir de support à un large spectre d'activités. Les mediaspaces sont sans doute les systèmes ayant couvert la plus grande partie de ce spectre. Et l'une des idées les plus intéressantes issues de ces travaux est de considérer le degré d'engagement comme une donnée variable que l'utilisateur peut explicitement ajuster. Le mediaspace RAVE proposait ainsi deux services qui était rigoureusement identiques sur le plan technique, *vphone* et *office share*, mais différents dans l'intention et dans le degré d'engagement attendu [Gaver *et al.*, 1992]. Le mediaspace Montage proposait lui un moyen d'ajuster le niveau d'engagement en permettant d'ajouter de l'audio en réponse à un coup d'œil vidéo, puis d'agrandir l'image du correspondant[13] [Tang et Rua, 1994].

Inspiré de manière plus ou moins consciente par ces travaux, j'ai moi aussi introduit une certaine variabilité du niveau d'engagement dans les systèmes sur lesquels j'ai travaillé et sur laquelle je vais m'arrêter un instant avant de proposer une approche plus générale.

2.4.1 Variabilité de l'engagement dans MirrorSpace, VideoProbe et VideoServer

La variabilité du degré d'engagement est évidente dans MirrorSpace, contrôlée par la distance au dispositif et représentée par un effet de flou proportionnel à cette distance. Cet effet ne suffit pas nécessairement à masquer tous les détails de l'activité d'une personne [Neustaedter *et al.*, 2006]. Mais cette personne sait que les autres savent qu'elle ne souhaite pas s'impliquer (et les autres le savent...). Le flou est en fait un moyen d'enrichir la communication vidéo pour indiquer sa volonté de rester en retrait, de réduire son engagement tout en gardant le contact. La connaissance commune de ce désir associée au fait que le système laisse malgré tout passer certains détails offre un espace de négociation qui fait cruellement défaut à la plupart des systèmes classiques. Le mode de calcul

[13] Une vidéo illustrant cette interaction est disponible à l'adresse `http://insitu.lri.fr/~roussel/digital-library/metadata/query/?q=montage`

de l'effet de flou détermine cet espace. En liant la force de l'effet appliqué à une image à la somme des distances mesurées localement et sur ce site, on permet à chacun de modifier dans une certaine mesure la représentation des autres, ce qui rend la négociation plus explicite.

Les transitions entre ses différents modes de fonctionnement jouent un rôle extrêmement important dans l'interaction avec VideoProbe. L'utilisation de la détection de mouvement permet de basculer simplement le système de la veille à un état dans lequel il est prêt à communiquer. Le retardateur de prise de vue déclenché par le mouvement constitue une offre répétée du système d'empêcher cette communication, représentée par le carré translucide qui recouvre progressivement l'image[14]. Il y a donc là encore possibilité de négociation du degré d'engagement. Le choix de ce degré n'est pas binaire, puisque l'activité des personnes détermine la fréquence de transmission des images. Bien que la fréquence maximale soit assez faible, de 10 à 15 images par seconde, VideoProbe a plusieurs fois été utilisé pour accompagner une conversation téléphonique, démontrant ainsi l'intérêt d'un système capable de proposer un degré d'engagement variable allant de la communication asynchrone à la communication synchrone ou quasi-synchrone.

L'intégration des services de communication de videoServer dans des documents HTML et courrier électroniques (Fig. 1.1, page 2) est un exemple d'association réussie et facilitant là encore les transitions entre des services synchrones et asynchrones. Plusieurs services vidéo peuvent être combinés en utilisant les URLs appropriées dans un même document. Quelques lignes de JavaScript permettent de mettre en œuvre une attention sélective, en transformant par exemple une image fixe en un flux vidéo au survol de la souris puis, à la manière de Montage, en un flux plus détaillé, plus rapide et affiché dans une nouvelle fenêtre lorsqu'on clique dessus. Les mécanismes de notification et de contrôle de VideoServer permettent également de définir différentes politiques d'accès par l'intermédiaire d'un langage de script. En fonction du contexte (e.g. identité de la personne distante, heure de la journée, document dans lequel doivent s'insérer les images), l'image capturée en temps réel peut ainsi être envoyée telle quelle, ou filtrée, mais aussi remplacée par une image préenregistrée (Fig. 2.15). À l'attention sélective offerte par la combinaison de services s'ajoute donc la notion d'exposition sélective.

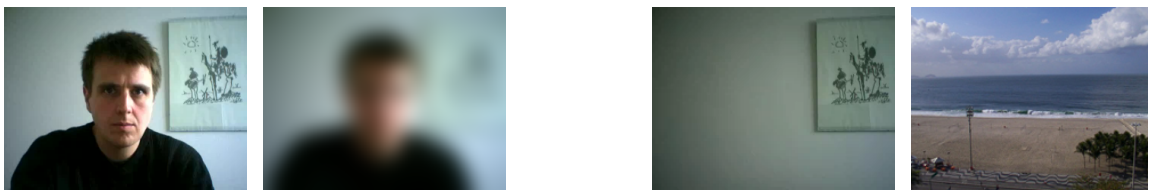


FIG. 2.15: Exemple d'exposition s'élective : image capturée, image filtrée, image préenregistrée ambiguë et indicateur explicite d'absence prolongée

[14]Initialement introduits pour ne pas encombrer l'interface et le disque dur, le vieillissement des photos suivi de leur disparition de l'interface procèdent d'une approche similaire : ils assurent un désengagement progressif, implicite et réversible jusqu'à un certain point.

2.4.2 Systèmes de communication multi-échelles

VideoServer facilite les transitions entre des communications textuelles asynchrones et vidéo synchrones et offre des mécanismes d'attention et d'exposition sélectives qui permettent d'ajuster le degré d'engagement. Mais la maîtrise de ces mécanismes impose certaines connaissances techniques. VideoProbe et MirrorSpace mettent en œuvre des moyens beaucoup plus directs et plus intuitifs pour ajuster le degré d'engagement. Ces systèmes préfigurent le type de recherche sur la communication médiatisée qu'il faut à mon avis mener. Je pense qu'il est en effet nécessaire de concevoir des systèmes permettant un degré d'engagement variable et des transitions fluides entre ces degrés. Un tel système permettrait de choisir simplement et rapidement le degré d'engagement le plus adapté à un contexte particulier. Il devrait également permettre de comprendre rapidement le degré d'engagement de chacune des personnes.

Comment un système de communication peut-il permettre différents degrés d'engagement ? Trois possibilités se dessinent à partir des exemples que nous venons de voir :

1. en permettant des transitions entre des services différents ;
2. en permettant de changer la fréquence des échanges ;
3. en permettant de changer le contenu des messages.

Dans le cas de la vidéo, le contenu peut être simplement modifié en jouant sur la taille ou les couleurs des images. Divers filtres spatiaux peuvent être appliqués pour les détériorer [Zhao et Stasko, 1998, Neustaedter *et al.*, 2006]. Des filtres temporels peuvent au contraire les enrichir en fournissant une indication de l'activité passée [Hudson et Smith, 1996, Gutwin, 2002]. Des filtres plus élaborés peuvent également éliminer certains détails [Coutaz *et al.*, 1998] tout en mettant en valeur certains autres [Karahalios et Donath, 2004, Chatting *et al.*, 2006]. Des techniques similaires ont été proposées pour l'audio [Smith et Hudson, 1995, Diaz-Marino et Greenberg, 2006]. Le cas du texte est plus simple dans la mesure où il est directement produit par l'utilisateur et ne résulte pas d'une capture extérieure.

L'expression *monde multi-échelle* a été proposée par Jul & Furnas [1998] pour décrire un monde dans lequel l'information existe à de multiples niveaux de détails. Le degré d'engagement tel que je le propose correspond à un niveau de détail jugé acceptable dans un contexte de communication donné. J'ai donc proposé d'utiliser l'expression *systèmes de communication multi-échelles* pour désigner les systèmes permettant un degré d'engagement variable [**Roussel et Gueddana, 2007**]. L'idée de transitions fluides renvoie à une variation continue ou du moins perçue comme telle de ce degré, i.e. du niveau de détail. En termes d'interfaces zoomables [Perlin et Fox, 1993], on s'intéresse ici à du *zoom continu*. La modification du contenu et la transition d'un service à un autre sont susceptibles de modifier le sens des messages échangés, et non seulement leur détail. On peut donc dans ce cas parler de *zoom sémantique*.

Comme dans le cas des interfaces zoomables, l'un des défis posés par les systèmes de communication multi-échelles réside dans la conception de techniques d'interaction ap-

propriées pour déclencher et contrôler les changements d'échelle, i.e. les transitions entre différents degrés d'engagement d'un même service ou entre différents services. L'objectif étant de réduire le travail articulatoire, ces techniques doivent être aussi directes et concises que possible. Dans ce contexte, comme on l'a vu avec VideoProbe et MirrorSpace, la caméra peut avantageusement jouer le rôle de dispositif d'entrée et d'autres capteurs peuvent s'avérer utiles. Un second défi consiste à concevoir des techniques de présentation permettant de distinguer facilement le degré d'engagement des personnes distantes.

En octobre 2005, j'ai obtenu de France Télécom R&D un contrat de recherche afin d'explorer ces problèmes dans le contexte d'environnements domestiques. J'assure depuis cette date la direction scientifique de la thèse de Sofiane Gueddana qui travaille avec moi sur ce sujet. Ensemble, nous avons notamment conçu et réalisé un premier système vidéo fondé sur l'approche multi-échelles : Pêle-Mêle.

2.4.3 Pêle-Mêle

Pêle-Mêle [**Gueddana et Roussel, 2006**] est un système de communication vidéo de groupe permettant un degré d'engagement variable allant de l'interaction synchrone fortement couplée à la communication asynchrone ou périphérique. Pêle-Mêle analyse en permanence l'activité des utilisateurs locaux et la classifie sur une échelle à trois niveaux : *absent*, *disponible* et *engagé*. L'activité observée en chacun des lieux reliés par le système détermine la nature de sa représentation à l'écran, qui combine potentiellement des images temps-réel et pré-enregistrées. L'affichage utilise une approche de type *focus-plus-context* reproduite à l'identique sur chaque site afin de faciliter le repérage mutuel.

Les images des lieux au niveau *engagé* apparaissent superposées au centre de l'écran tandis que celles des lieux aux niveaux *disponible* et *absent* sont visibles en périphérie (Fig. 2.16). Une animation permet la transition entre ces représentations. Les images des activités passées au niveau *engagé* sont également affichées le long d'une perspective temporelle : elles rétrécissent lentement et dérivent en profondeur vers le centre de l'écran avec le temps. Le résultat à l'écran est visuellement proche de l'affichage d'IMVis [Neustaedter *et al.*, 2002], un système destiné à visualiser la disponibilité d'un ensemble de personnes connectées à travers MSN Messenger. L'utilisation des trois axes offerts par la vue 2D+perspective est toutefois différente[15].

En fonction du niveau d'activité détecté, différents filtres spatiaux ou temporels sont appliqués sur les images. Au niveau *engagé*, elles sont affichées telles quelles. A ce niveau, l'association d'une communication audio est également possible à travers un logiciel de téléphonie sur IP ou un téléphone mobile. Au niveau *disponible*, les images sont retar-

[15]IMVis utilise la perspective pour indiquer à la fois la disponibilité, l'intérêt porté à une personne et l'historique de sa conversation. Mais il laisse les deux autres dimensions à la libre interprétation de l'utilisateur.

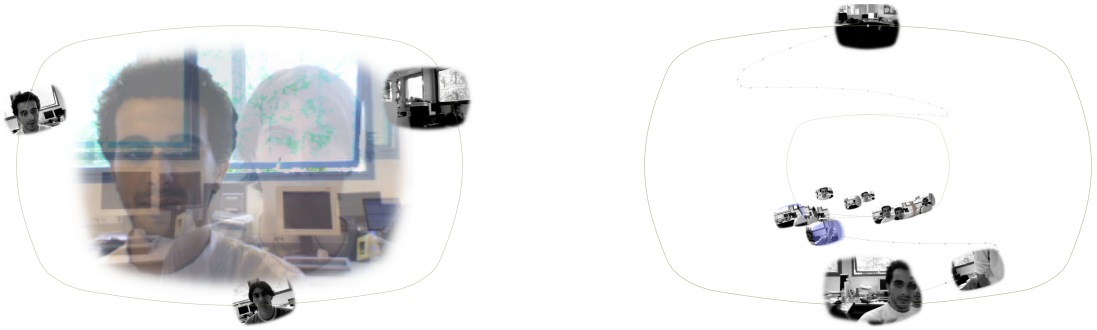


FIG. 2.16: Exemples d'affichages de Pêle-Mêle

dées, présentées en niveaux de gris et composées à des images récentes du lieu concerné (Fig. 2.17, images de gauche). Au niveau *absent*, l'image affichée est la dernière transmise à l'état *disponible*, filtrée en fonction de son âge à l'aide d'un effet de type "peinture à l'huile" (Fig. 2.17, images de droite). Lorsqu'aucun des sites ne se situe au niveau *engagé*, des clips précédemment enregistrés à ce niveau sont joués de manière aléatoire.

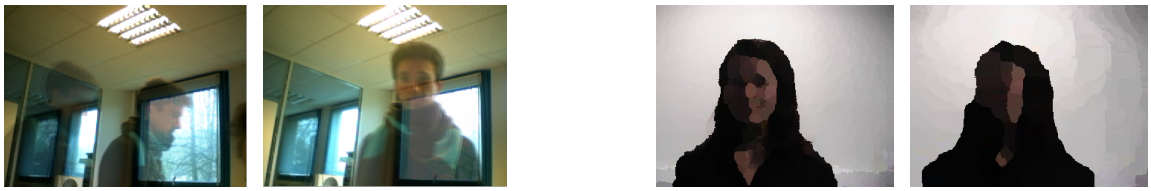


FIG. 2.17: Exemples d'enrichissement temporel et de dégradation spatiale des images

Les commentaires recueillis lors de présentations informelles indiquent que les utilisateurs comprennent rapidement le fonctionnement du détecteur d'activité et son impact sur le système. Plusieurs l'ont toutefois jugé trop sensible, regrettant qu'un léger déplacement suffise dans certains cas à causer d'importantes perturbations de l'affichage. Un autre problème fréquemment cité est la difficulté de se maintenir à un niveau particulier. Mais même si elle est parfois frustrante, l'interaction est néanmoins généralement appréciée des utilisateurs. L'organisation visuelle de l'interface est plus difficile à comprendre. La signification des trajectoires en perspective n'est pas toujours évidente pour les utilisateurs. Une fois expliqué, l'intérêt de ces trajectoires est toutefois compris. Les nombreux déplacements et changements de taille des images actuelles ou passées ont souvent été critiqués. De nombreux utilisateurs trouvent par ailleurs les images du passé mal exploitées. Enfin, la complexité de la scène rend pour certain difficile la communication synchrone, face à face.

La première version de Pêle-Mêle relevait un peu de l'exercice de style : il s'agissait d'essayer d'appliquer les principes liés à l'approche multi-échelles dans un unique système ne proposant qu'une unique vue. Si certaines réactions des utilisateurs sont assez critiques, d'autres sont plutôt encourageantes et un certain nombre de suggestions ont été faites. Une nouvelle version du système est actuellement en cours de développement afin

de prendre en compte ces divers éléments.

Chapitre

3 Nouvelles interactions homme-machine

Parallèlement à mes travaux sur la communication médiatisée, je m'intéresse depuis de nombreuses années à l'évolution des systèmes graphiques interactifs. Cet intérêt s'est focalisé sur le thème du *bureau informatique* durant la fin de ma thèse et au cours de mes séjours post-doctoraux, alors que je développais des prototypes qui utilisaient son image capturée en temps réel (Fig. 1.3, page 3). Du bureau, mon intérêt s'est rapidement déplacé aux fenêtres lorsque j'ai commencé à découper les images du premier pour en extraire celles des secondes. Mon attention s'est ensuite logiquement portée sur la gestion de fenêtre et les systèmes de fenêtrage. Petit à petit, je me suis intéressé à ce que l'on pourrait appeler *l'interaction au quotidien* entre un utilisateur et ses données, à travers le système informatique et ses applications.

La majorité des concepts et techniques utilisés pour l'interaction au quotidien ont été conçus pour le Xerox Star dans la deuxième moitié des années 1970. Les interfaces des systèmes actuels ressemblent d'ailleurs beaucoup à celle de cet ancêtre lointain (Fig. 3.1). De nombreuses choses ont pourtant changé au cours des trente dernières années. Les capacités et performances du matériel d'aujourd'hui sont sans rapport avec celles du Star, les usages de l'informatique sont plus divers et la quantité de données manipulée est bien plus grande. On peut dès lors s'interroger : la métaphore et les techniques d'interaction proposées par le Star sont-elles toujours pertinentes ?

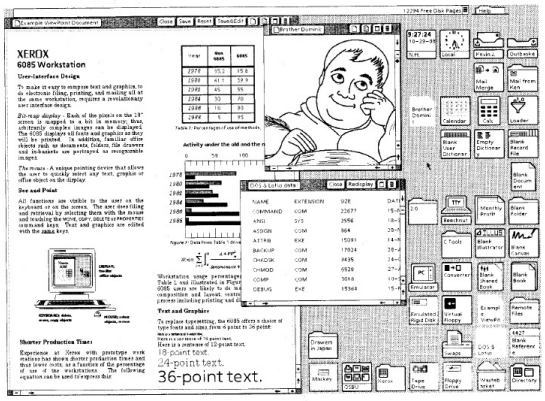


Fig. 3.1: Interface graphique du Star [Johnson *et al.*, 1989]

3.1 Problématique

La conception du Star fut guidée par huit principes [Smith *et al.*, 1982]. **Le premier** est que le système doit être conçu selon un modèle conceptuel familier de ses utilisateurs. Le modèle retenu repose sur la métaphore du bureau, au sens "endroit où l'on travaille". Dans ce bureau, les concepteurs du Star créent toute une série d'objets analogues à ceux que l'on trouve dans un bureau réel : du papier, des documents (fichiers), des dossiers (répertoires), des armoires (disques durs), des boîtes aux lettres, etc. Ils créent aussi un bureau, au sens "table de travail" cette fois, qui occupe l'arrière-plan de l'écran. **Le deuxième** principe de conception est que tous les objets et commandes relatifs à la tâche en cours doivent être visibles à l'écran. Icônes et fenêtres sont ainsi utilisées pour représenter et manipuler les documents sur le bureau. **Le troisième** principe dit que la vision que l'on a des documents à l'écran doit être conforme à ce qui sera imprimé. C'est le fameux *WYSIWYG*[1]. Selon **les quatrième, cinquième, sixième et septième** principes, le système doit proposer des commandes génériques applicables dans différents contextes, être cohérent, simple et éviter les modes. **Le dernier** principe est que le système doit pouvoir être adapté ou étendu par l'utilisateur.

L'une des caractéristiques essentielles du Star liée à la métaphore du bureau est qu'il propose un environnement centré sur la notion de document. Un nouveau document est créé à partir de papier vierge, ou d'un document existant. Il peut contenir du texte, des dessins, des formules mathématiques, des tableaux, tous éditables sur place : pour l'utilisateur, la notion d'application est inexistante. C'est sans doute là la principale différence avec les environnements actuels, centrés sur des applications autonomes, indépendantes les unes des autres, faciles à commercialiser [Beaudouin-Lafon, 2007]. Et cette différence explique sans doute le fait que l'environnement interactif dans son ensemble et la métaphore sur laquelle il est basé ont reçu assez peu d'attention et n'ont pas réellement évolué.

3.1.1 Un bureau encombré

La mise en œuvre de la métaphore du bureau repose en grande partie sur l'utilisation des fenêtres. L'origine de ce concept remonte au années 1960 [Engelbart, 1962, Sutherland, 1963, Kay, 1968]. Les premières fenêtres graphiques superposables telles que nous les connaissons furent créées au début des années 1970 au Xerox PARC pour l'environnement Smalltalk [Kay, 1993]. Elles furent ensuite utilisées par le Star. Mais malgré toutes ses qualités, celui-ci fut un échec commercial et il fallut attendre 1984 pour que le concept atteigne le grand public à travers le Macintosh d'Apple. L'usage des fenêtres se généralisa par la suite à travers les systèmes X Window et Microsoft Windows.

[1] *What You See Is What You Get*

La diversité et le nombre croissant de nos activités informatiques font qu'il est de plus en plus difficile de les organiser. Ces activités sont souvent menées en parallèle, entraînant de fréquents changements de contexte. Elles reposent également souvent sur de multiples documents qui sont manipulés à travers des applications aux nombreuses barres d'outils et palettes. Depuis le Star, le nombre de fenêtres présentes à l'écran a ainsi considérablement augmenté. Mais du point de vue de l'interaction avec ces fenêtres, les systèmes actuels ne diffèrent du Star que sur des détails mineurs. Les techniques proposées sont toujours les mêmes. Les progrès de l'informatique graphique offrent pourtant de nouvelles possibilités. Les modèles graphiques actuels sont plus riches, les écrans sont plus grands, ils ont une meilleure résolution et les cartes graphiques sont plus performantes. Mais ces progrès ne servent qu'à quelques applications. Le système de fenêtrage n'en tire pas vraiment parti.

Peu de travaux ont été consacrés à l'étude des différentes pratiques en matière de gestion de fenêtres [Hutchings et Stasko, 2004a]. Parmi les nouvelles techniques d'interaction proposées, rares sont celles qui ont été formellement évaluées, la plupart ayant été conçues selon une approche basse fidélité qui rend impossible toute étude longitudinale [Bell et Feiner, 2000, Beaudouin-Lafon, 2001, Hutchings et Stasko, 2004b, Dragicevic, 2004]. Très peu ont été intégrées à de réels systèmes de fenêtrages. Task Gallery [Robertson *et al.*, 2000] permettait bien d'intégrer des applications Windows existantes dans un environnement 3D, mais les modifications de Windows 2000 nécessaires à son fonctionnement ne furent jamais rendues publiques.

FIG. 3.2: Fold n' Drop [Dragicevic, 2004] et Task Gallery [Robertson *et al.*, 2000]

Plusieurs travaux montrent 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 [Robertson *et al.*, 2004, Hutchings et Stasko, 2005, Tan *et al.*, 2004]. Mais les systèmes de fenêtrage des systèmes d'Apple et de Microsoft sont fermés et la complexité du système X Window rebute la plupart des chercheurs.

Certaines initiatives comme Looking Glass[2] ou Croquet[3] permettent l'intégration d'applications X Window dans des environnements graphiques expérimentaux, mais ces environnements n'implémentent qu'une partie des protocoles relatifs à la gestion de fenêtres, ce qui les rend difficilement utilisables au quotidien et donc difficilement évaluables.

Un exemple illustre à la fois l'impact que peuvent avoir des recherches sur la gestion de fenêtres et le temps nécessaire pour cela. Rooms, un gestionnaire de fenêtres intégrant le concept d'espaces de travail virtuels, fut proposés en 1985 par Card et Henderson afin de palier aux problèmes posés par la petite taille du bureau électronique en comparaison des espaces de travail physiques habituels [Card *et al.*, 1984, Henderson et Card, 1986]. Initialement implémenté sur des machines Interlisp-D de Xerox, Rooms fut porté sur le système X Window en 1989. Diverses applications ont par la suite adapté le concept aux systèmes de Microsoft et d'Apple. Mais il aura fallut attendre Leopard (Mac OS X 10.5.1) pour qu'Apple le propose en standard[4].

3.1.2 Des interfaçes peu adaptées et difficilement adaptables

Les évolutions successives des applications s'accompagnent généralement d'une augmentation importante du nombre de leurs commandes [Beaudouin-Lafon, 1997]. Bien que nous n'en utilisions qu'un petit sous-ensemble [McGrenere *et al.*, 2002], nombre d'entre elles sont disponibles en permanence à l'écran, augmentant inutilement la taille des menus, barres d'outils et palettes et réduisant ainsi l'espace de travail disponible. Ces interfaces envahissantes peuvent être le fait de programmeurs insuffisamment sensibilisés aux problèmes d'interaction homme-machine. Mais le fait est qu'il est difficile de prévoir la manière dont un logiciel va effectivement être utilisé, surtout lorsqu'il est diffusé à grande échelle.

Une façon de résoudre ce problème est de permettre à l'application ou à l'utilisateur de modifier l'interface. On parle dans le premier cas d'*adaptativité* [Kühme, 1993], et d'*adaptabilité* [Kantorowitz et Sudarsky, 1989] dans le second. Les interfaces adaptatives modifient leur apparence en fonction d'un algorithme prédéfini. Elles sont souvent critiquées pour l'effet de surprise qu'elles peuvent provoquer – en supprimant des éléments d'un menu par exemple – et le fait qu'elle rendent plus difficile la formation d'habitudes. Les interfaces adaptables laissent au contraire l'initiative à l'utilisateur. Elles nécessitent toutefois l'ajout d'interfaces et/ou techniques d'interaction secondaires pour permettre leur personnalisation. De nombreuses applications permettent par exemple de modifier leurs menus, barres d'outils et palettes par des opérations de glisser-déposer. Mais les interfaces secondaires proposées sont souvent de bien piêtre qualité (Fig. 3.3).

[2] `http://wwws.sun.com/software/looking_glass/`
[3] `http://www.opencroquet.org/`
[4] `http://www.apple.com/macosx/leopard/features/spaces.html`

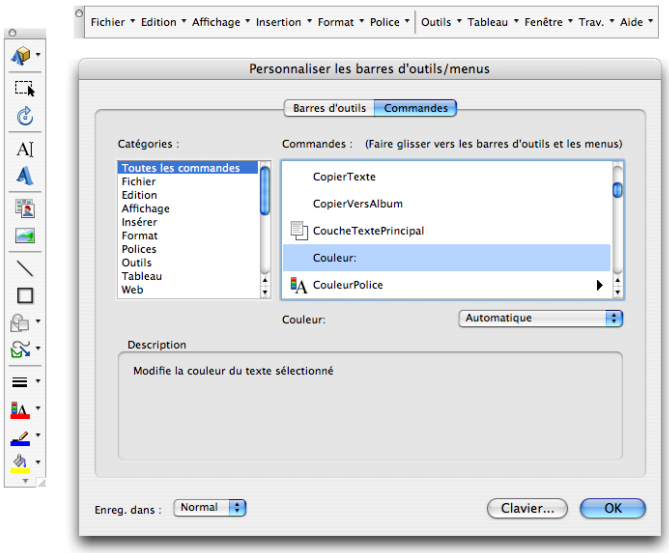


FIG. 3.3: Interface de personnalisation des menus et barres d'outils de Microsoft Word 2004. La liste de droite montre les commandes d'une des 22 catégories proposées dans la liste de gauche. Plus de 1100 commandes sont ainsi disponibles. Elles peuvent être ajoutées ou supprimées des menus et barres par glisser-déposer, mais ces opérations ne peuvent pas être annulées. Les commandes présentes dans les menus et barres sont toujours visibles dans la liste. La fenêtre fait environ 600x500 pixels. Elle peut être déplacée mais pas redimensionnée.

3.1.3 Des armoires qui débordent

Les disques durs de nos ordinateurs contiennent de plus en plus de données. A titre d'exemple, mon propre répertoire contient aujourd'hui près de 220.000 fichiers, dont environ 8.200 documents texte, 41.000 images (illustrations ou photos), 1.800 vidéos, 9.000 morceaux de musique, 700 présentations et plus de 42.000 courriers électroniques[5]. A l'échelle des capacités cognitives humaines, ces chiffres correspondent à des masses de données considérables, d'autant que chacun de ces fichiers contient lui-même une grande quantité d'information : mots, paragraphes, pages d'un document texte, clips d'une vidéo, transparents d'une présentation, etc. La manipulation de ces données implique donc de fréquentes tâches de recherche d'un élément parmi quelques dizaines de milliers à quelques dizaines de millions, selon la granularité de l'élément recherché.

La taille des disques durs ne cesse d'augmenter. Les machines sont couramment vendues avec des disques de plusieurs centaines de giga-octets. Selon certaines études, le prix de l'unité de stockage est aujourd'hui tel qu'il est devenu inutile de supprimer des fichiers pour "faire de la place" [Gemmell *et al.*, 2002] : en rachetant un disque de temps en temps, nous devrions être capables de conserver tout ce qui nous passe entre les mains et que nous jugeons digne d'intérêt. Toutefois, comme nous l'avons vu, la représentation

[5]Ces chiffres sont approximatifs, le calcul se faisant à partir d'heuristiques basées sur le nom des fichiers

iconique des fichiers et répertoires ainsi que la plupart des techniques d'interaction que nous utilisons pour naviguer dans nos données ont été conçues pour le Star, une machine dont la capacité maximale du disque dur était de 40 mégaoctets. Et nous observons chaque jour que ces représentations et ces techniques ne sont plus adaptées à la quantité et à la diversité des données personnelles et professionnelles que nous avons à gérer.

Plusieurs équipes ont proposé des systèmes d'indexation par le contenu et de recherche spécifiquement conçues pour un usage quotidien par le grand public sur ses propres données [Gemmell *et al.*, 2002, Dumais *et al.*, 2003, Quan *et al.*, 2003]. Plusieurs systèmes commerciaux de ce type ont également fait leur apparition, tels que Google Desktop Search ou Apple Spotlight. La plupart de ces systèmes sont issus de travaux antérieurs sur la recherche d'information dans des bases de données ou sur le Web. On peut cependant douter que cette approche soit la plus efficace pour ce qui concerne les disques durs de nos machines. D'une part, s'agissant d'une collection de données hétérogène, il n'est pas toujours évident d'exprimer le critère de recherche sous une forme utilisable par le système, notamment si l'on s'intéresse à des données multimédia (e.g. photos, vidéos). Mais surtout, ce type d'approche ignore totalement un point crucial qui différencie les données d'un utilisateur de celles du Web : il les a produites, reçues d'une autre personne ou téléchargées puis éventuellement consultées ou modifiées. Il existe donc une histoire qui le relie à ces données et en ce sens, elles lui sont donc familières.

3.1.4 Résumé des questions abordées

Trente ans après, les huit principes de base du Star restent globalement de bons principes. Mais leur traduction concrète dans les systèmes actuels mérite sans doute d'être revue. Les commandes génériques pourraient être plus utilisées, par exemple (principe #4). De nombreuses interactions modales devraient être supprimées (principe #7). Et les mécanismes permettant aux utilisateurs d'adapter ou d'étendre les interfaces pourraient être améliorés (principe #8). Le deuxième principe – *tous les objets et commandes relatifs à la tâche en cours doivent être visibles* – est plus délicat à prendre en compte : la notion de relativité à la tâche est assez vague et on l'a vu, le nombre d'objets et de commandes a considérablement augmenté. Mais le principe le plus difficile à appliquer aujourd'hui est sans doute le premier.

La métaphore du bureau, au sens "endroit où l'on travaille", fut certainement utile pendant de nombreuses années, permettant à des utilisateurs novices d'utiliser l'informatique de manière efficace. Mais le nombre de nos activités informatiques et la quantité de données auxquelles nous avons accès ont tous deux changé d'échelle. Les techniques adaptées à la gestion d'un modeste bureau ne sont pas nécessairement adaptées à la gestion d'une bibliothèque... surtout dans le cas d'utilisateurs novices. Depuis plusieurs années, je m'intéresse donc aux moyens de faire évoluer la métaphore du bureau. Je ne cherche pas nécessairement à la remplacer, mais plutôt à améliorer, à étendre certains de ses composants.

Dans ce contexte, les questions auxquelles je me suis intéressé sont les suivantes :

Comment faire évoluer le bureau ?

Beaudouin-Lafon et Lassen [2000] ont montré que des techniques d'interaction avancées telles que les *marking menus* [Kurtenbach et Buxton, 1994], les *toolglass* [Bier *et al.*, 1993] ou d'autres techniques bi-manuelles pouvaient être avantageusement combinées au sein d'une même application. Il est cependant frustrant de constater que ces techniques restent inaccessibles au niveau du bureau. Comme je l'ai dit, de nombreuses autres techniques ont été spécifiquement proposées pour la gestion de fenêtres mais sont restées au stade de prototypes. En résumé, les nouvelles idées ne manquent pas. Mais la mise en œuvre de ces idées est délicate, ce qui complique leur évaluation et empêche leur diffusion.

Comme le dit l'expression, le diable est dans les détails, et ceux-ci ne sont généralement pas visibles dans de simples prototypes. Evaluer les nouvelles techniques d'interaction pour le bureau de manière crédible, même informelle, nécessite de pouvoir les tester dans un contexte d'utilisation réel. Mais l'implémentation d'un système de fenêtrage complet est une tâche que peu de personnes peuvent accomplir. Et les systèmes existants sont inaccessibles (Windows ou Mac OS X) ou complexe à modifier car reposant sur un ensemble d'extensions plus ou moins stables accumulées au fil des ans (X Window).

Comment implémenter un gestionnaire de fenêtres zoomable ou destiné à une table interactive, par exemple ? Tel est le type de question qui a motivé mes premiers travaux sur l'evolution du bureau.

Comment faciliter la gestion des données ?

Le deuxième principe de conception du Star est souvent résumé par l'expression "voir et pointer plutôt que se souvenir et taper". Nous sommes aujourd'hui à un point où il y a beaucoup trop de choses à voir et où dans certains cas, il est plus rapide de se souvenir que de chercher à l'écran. Cela est particulièrement vrai pour les données que nous avons nous même produites, collectées ou reçues d'autres personnes. Au cours des dernières années, je me suis intéressé à la notion d'historique de l'interaction et à ses usages potentiels pour faciliter la gestion de ces données que j'ai qualifié de *familières*. Cet intérêt s'est porté sur les données enregistrées dans des fichiers, ainsi que sur celles accessibles sur le Web.

Comment faciliter l'accès aux données, comment les rendre utilisables tout en réduisant au minimum l'effort d'organisation préalable nécessaire ? Telles sont les questions qui ont motivés ces travaux.

Mes travaux sur ces questions se sont en partie déroulés dans le cadre d'un projet financé par l'ACI Masses de Données et ont fait l'objet de huit publications dont je suis l'auteur ou le coauteur :

- CLIHC 2003 : [**Roussel, 2003**]
- IHM 2005 : [**Roussel *et al.*, 2005**] et [**Roussel et Chapuis, 2005**] (articles courts)

– ACM UIST 2005 : **[Chapuis et Roussel, 2005]**
– ACM UIST 2006 : **[Stuerzlinger *et al.*, 2006]**
– ACM CHI 2007 : **[Chapuis et Roussel, 2007]** et **[Tabard *et al.*, 2007]**
– IHM 2007 **[Besacier *et al.*, 2007]** (démonstration)

Ces travaux m'ont également donné l'occasion de participer à un atelier de la conférence WWW **[Roussel *et al.*, 2006b]**.

La suite de ce chapitre en présente un résumé.

3.2 Outils et techniques d'interaction pour un nouveau bureau informatique

En avril 2000, alors que je développais un premier prototype capable d'extraire les images d'applications de celle d'un bureau et de les recomposer (Fig. 3.4), une équipe de Microsoft Research présenta à la conférence ACM CHI un système au principe similaire : Task Gallery [Robertson *et al.*, 2000]. Impressionné par l'apparente robustesse de ce système et la composition de l'équipe qui l'avait créé, je décidai de me consacrer à d'autres sujets et de ne plus apporter que des modifications mineures à mon prototype. Deux ans plus tard toutefois, étonné et déçu que personne n'ait donné de suite à Task Gallery – y compris chez Microsoft, je ressortis mon prototype des cartons et décidai d'en faire la publicité sous le nom de VideoWorkspace **[Roussel, 2002c]**. L'accueil reçu par ce prototype à IHM 2002 fut bon mais un peu court, mon exposé étant le dernier de la conférence... Quelques modifications plus tard, je décidai donc de tenter ma chance ailleurs et sous un autre nom, plus neutre : Ametista **[Roussel, 2003]**. L'accueil reçu par cette nouvelle version à CLIHC 2003 fut bon, mais un peu court, mon exposé étant à nouveau le dernier de la conférence...

FIG. 3.4: Prototypes de bureau réalisés en 2000, 2001 et 2002

Ametista permettait de prototyper de nouvelles techniques d'interaction pour la gestion de fenêtres et de tester ces techniques avec de réelles applications non modifiées. Mais il dépendait d'un serveur Xvnc [Richardson *et al.*, 1998] qui ne savait rien du processus de composition, ce qui rendait certaines choses inutilement compliquées et d'autres choses impossibles. Ametista n'implémentait également qu'une infime partie des pro-

tocoles standards liés à la gestion de fenêtres, ce qui le rendait difficilement utilisable au quotidien. Une solution existait à ces problèmes : il "suffisait" de remplacer Xvnc par un serveur X modifié de manière spécifique et d'intégrer au système un véritable gestionnaire de fenêtres. Mais le travail à effectuer était titanesque.

Le 22 septembre 2003, un miracle se produisit : un mathématicien du nom d'Olivier Chapuis, chercheur au CNRS, me proposa de travailler avec moi à l'amélioration d'Ametista. Deux mois plus tard, Olivier avait implémenté la première version du nouveau système : Metisse [**Chapuis et Roussel, 2005**, **Roussel et Chapuis, 2005**]. Ce système fut rendu public en janvier 2004 et Olivier n'a cessé de l'utiliser quotidiennement depuis. Cet usage quotidien nous a progressivement ammené à modifier le système, à l'étendre [**Stuerzlinger *et al.*, 2006**, **Chapuis et Roussel, 2007**] et a repenser plusieurs fois sa conception. La suite de cette section présente l'ensemble des travaux réalisés dans ce contexte.

3.2.1 Metisse

Metisse [**Chapuis et Roussel, 2005**, **Roussel et Chapuis, 2005**] est un nouveau système de fenêtrage conçu pour répondre à deux objectifs précis : faciliter la mise en œuvre de nouvelles techniques de gestion de fenêtres, et permettre leur évaluation en les rendant utilisables au quotidien dans un environnement applicatif standard. Metisse n'est pas destiné à l'étude d'un style particulier d'interaction mais doit plutôt être vu comme une plate-forme permettant l'exploration de nouveaux environnements interactifs capables d'intégrer des applications traditionnelles.

Metisse repose sur le découplage du rendu des fenêtres des applications et de la composition interactive des images ainsi produites. Il se veut être un système de fenêtrage complet et compatible avec les applications existantes. Le cœur du système, Xmetisse, est un serveur X Window 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 modifiée du gestionnaire de fenêtres FVWM[6] associée à un module interactif spécifique, FvwmCompositor (Figure 3.5). L'ensemble est implémenté en C et C++ sur les plate-formes Linux et OS X.

Metisse vient pour le moment se greffer au-dessus du système graphique de la plate-forme sur laquelle il tourne. FvwmCompositor est une application OpenGL qui utilise le système natif pour afficher la composition des images rendues par Xmetisse. Lorsqu'une application redessine l'une de ses fenêtres, Xmetisse le notifie à FvwmCompositor en précisant la région modifiée à l'aide d'un protocole spécifique similaire à celui de VNC [Richardson *et al.*, 1998]. D'autres notifications transitent par ce protocole, pour indiquer par exemple la création et la destruction de fenêtres, ou les changements de curseur. Il permet également à FvwmCompositor de faire suivre à Xmetisse les événe-

[6] `http://www.fvwm.org/`

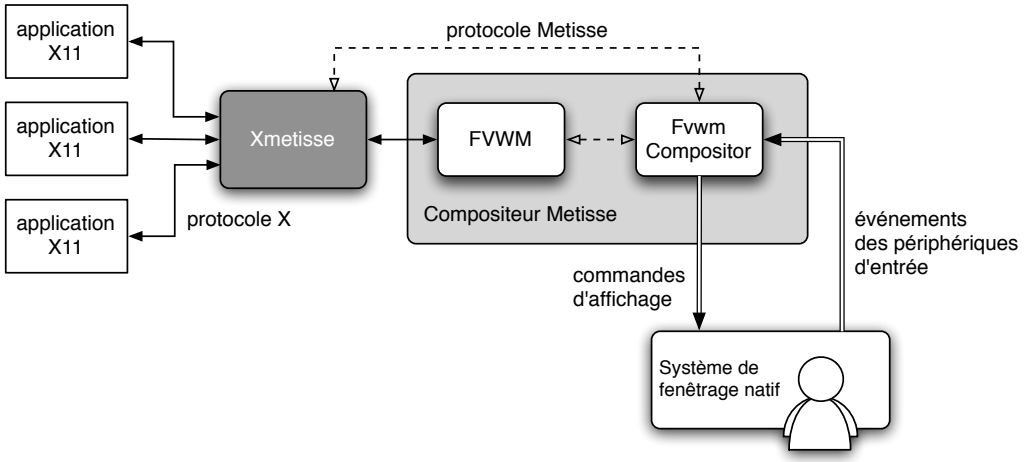


FIG. 3.5: Architecture du système Metisse

ments des périphériques d'entrée en les transformant éventuellement au passage. Les opérations classiques sur les fenêtres (e.g. déplacement, redimensionnement, iconification) sont gérées par FVWM, déclenchées par les événements des périphériques d'entrée reçus de Xmetisse. L'ajout de nouvelles techniques d'interaction peut se faire à ce niveau, à travers un langage de script, ou par modification du code C++ de FvwmCompositor.

Xmetisse diffère des serveurs X traditionnels sur plusieurs points. La gestion des événements, par exemple, est particulière. Lorsqu'un serveur X traditionnel reçoit un événement souris, il utilise la position du pointeur et sa connaissance du placement des fenêtres pour identifier celle devant le recevoir. Dans le cas de Metisse, le compositeur étant libre de transformer les images des fenêtres (e.g. par translation, rotation et changement d'échelle), les événements souris relayés vers le serveur doivent nécessairement spécifier la fenêtre concernée. Pour la même raison, le compositeur est également responsable de l'envoi d'événements *Enter* et *Leave* aux fenêtres concernées lors des déplacements du pointeur.

Xmetisse est toutefois un véritable serveur X dérivé de Xserver[7] et compatible avec toutes les applications X Window existantes. Outre les opérations classiques déjà évoquées, FVWM sait également gérer d'autres aspects plus complexes de la gestion de fenêtre, comme les bureaux virtuels, et implémente les différents protocoles qui s'y rapportent (e.g. ICCCM, EWMH). Dans le cas où FvwmCompositor n'applique pas de transformation particulière, Metisse se comporte donc comme un environnement X Window tout à fait standard. L'association de Xmetisse, FVWM et FvwmCompositor fournit donc un système de fenêtrage performant, robuste, compatible avec les standards existants et donc utilisable – et utilisé – au quotidien.

FvwmCompositor implémente plusieurs nouvelles opérations élémentaires sur les fenêtres, telles que le changement d'échelle, la rotation en trois dimensions, ou le chan-

[7] `http://freedesktop.org/wiki/Software/Xserver`

gement d'opacité et de luminosité. FVWM permet d'associer à l'aide de scripts ces opérations à des événements clavier/souris (e.g. une combinaison de touches particulière) ou de plus haut niveau (e.g. la mise en arrière plan d'une fenêtre). Lorsqu'un déplacement est initié, la fenêtre concernée est 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. 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 (Figure 3.6, image de gauche).

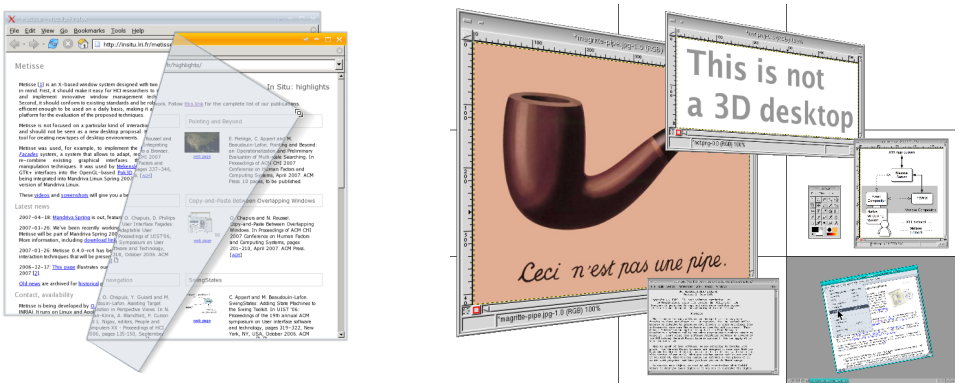


FIG. 3.6: Métaphore papier augmentée permettant d'accéder rapidement au contenu d'une fenêtre partiellement masquée et bureau 3D zoomable

Metisse permet également d'implémenter des opérations s'appliquant à un groupe ou à l'ensemble des fenêtres. La configuration standard du système intègre ainsi 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 hors de l'écran physique au-delà d'un certain seuil, une transition de la vue sur l'écran virtuel est automatiquement déclenchée. 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. L'utilisateur peut alors toujours interagir librement avec les fenêtres (Figure 3.6, image de droite). Il peut également déclencher un zoom automatique sur une des neufs parties de l'écran virtuel d'un simple clic.

Plusieurs configurations de Metisse ont été développées qui transforment automatiquement les fenêtres en fonction de leur position. L'une d'entre elles réduit ainsi l'échelle des fenêtres lorsque qu'elles sont approchées du bord de l'écran (Figure 3.7, image de gauche), les empêchant d'en sortir et les conservant donc à portée de vue et de souris à la manière de Scalable Fabric [Robertson *et al.*, 2004]. Il fut là encore 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. Une autre configuration est destinée à un usage du système sur une table interactive [Dietz et Leigh, 2001]. Dans cette configuration, les fenêtres sont automatiquement pivotées lorsqu'elles sont déplacées pour être toujours orientées dans le sens de lecture correspondant au bord de la

table le plus proche (Figure 3.7, image de droite).

Metisse permet de dupliquer une fenêtre, la copie étant ensuite manipulable comme l'original. Dans la configuration précédente, cette possibilité permet à deux utilisateurs se faisant face de voir un même document. Bien que le système sache gérer de multiples périphériques d'entrée, rien n'est encore prévu pour gérer les conflits pouvant survenir du fait de leur utilisation simultanée sur une même fenêtre. Indépendamment des usages collaboratifs qui pourraient en découler, la duplication de fenêtre s'avère toutefois très utile dans le cas de l'utilisation de grandes surfaces d'affichage (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 dans d'autres fenêtres, une possibilité qui sera développée dans la section suivante.

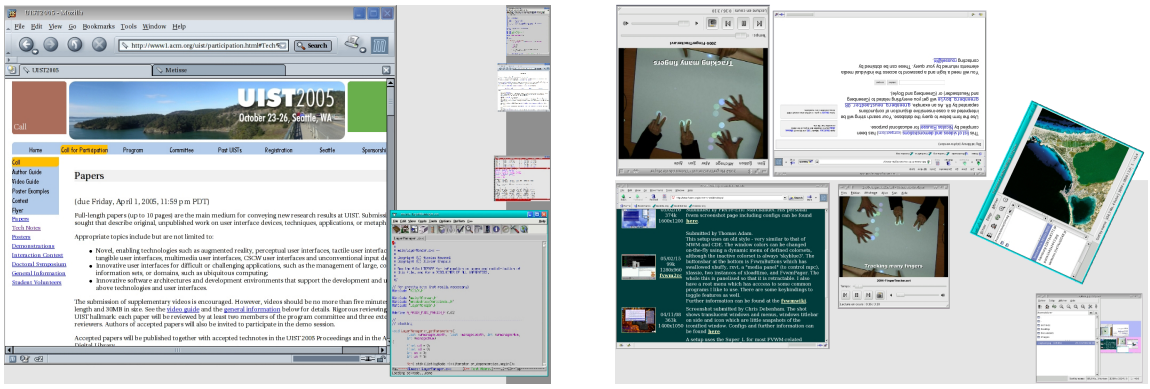


FIG. 3.7: Exemples de transformations géométriques dépendantes de la position des fenêtres

La nette séparation entre le serveur et le compositeur de Metisse rend possibles de nombreux effets ou transformations graphiques déjà présents dans les systèmes de fenêtrage d'Apple et Microsoft et dans certaines configurations X Window basées sur Compiz[8]. Metisse est également à rapprocher du projet Looking Glass[9] de Sun, un prototype de bureau 3D en Java dans lequel peuvent tourner des applications X Window. Les différences entre Metisse et ces systèmes peuvent se résumer ainsi :

– A la différence des systèmes propriétaires et fermés d'Apple et Microsoft, Metisse et les autres systèmes basés sur X Window sont ouverts, propres à l'expérimentation de nouvelles techniques d'interaction.
– Contrairement à Looking Glass, Metisse n'impose aucune contrainte sur le style d'interaction et le langage de programmation à utiliser pour la composition. Le couple FVWM+FvwmCompositor n'est qu'un exemple. Le serveur ne faisant pas de supposition particulière sur le processus de composition, d'autres applications peuvent s'y connecter. Notre souhait est que soient développés de nombreux compositeurs adap-

[8] http://www.opencompositing.org/
[9] http://wwws.sun.com/software/looking_glass/

tés à des problématiques spécifiques. Un squelette minimum pouvant servir de base a été développé dans cette optique.
– Le modèle de gestion des événements de Metisse permet à travers ses différentes redirections la mise en œuvre de techniques d'interaction impossibles avec les approches X classiques, comme la duplication de fenêtres. Les développeurs de Looking Glass et Compiz semblent conscients de cet avantage mais n'arrivent pas, pour le moment, à se mettre d'accord avec les développeurs de serveurs X pour faire rapidement les changements nécessaires.
– De part son architecture et l'utilisation d'un protocole orienté bitmap similaire à celui de VNC, nous pensons que Metisse est mieux adapté à de futures applications qui mettront en jeu différents compositeurs pour des usages collaboratifs et/ou multi-surfaces (e.g. utilisation couplée d'un PDA ou d'un ordinateur portable avec un mur d'images). En contrepartie, il est clair que l'architecture et le protocole actuels introduisent une dégradation de performances pour les applications qui bénéficient habituellement d'une accélération matérielle du rendu (e.g. lecteurs multimédia ou applications OpenGL).

Metisse est ou a été utilisé par plusieurs groupes extérieurs à In Situ. Il a été utilisé par les créateurs du jeu Pok3D[10] pour intégrer des widgets 2D réalisés avec GTK+ dans un graphe de scène 3D affiché avec OpenGL. Il s'agit à ma connaissance de la première utilisation d'une approche de type composition pour ce genre d'intégration, les solutions habituelles reposant sur l'utilisation de widgets décrits et programmés comme des objets 3D notoirement inférieurs à ceux de GTK+. L'équipe-projet ALCOVE de l'INRIA développe actuellement un module compositeur Metisse pour son environnement collaboratif Spin|3D [Dumas *et al.*, 1999] afin de pouvoir répondre à la demande d'utilisateurs souhaitant y retrouver leurs applications habituelles (e.g. traitement de texte, navigateur web). Je participe enfin à l'encadrement de la thèse de Guillaume Besacier qui travaille sur les tables interactives et qui a dans ce contexte développé un compositeur Metisse pour l'environnement DiamondSpin [Shen *et al.*, 2004, **Besacier *et al.*, 2007**].

Le code source de Metisse est disponible gratuitement sous licence GPL sur son site Web[11] qui présente également de nombreuses vidéos illustrant les nouvelles interactions qu'il rend possible. Distribué début 2007 sous forme de "Live CD" de démonstration par la société Mandriva, il fait aujourd'hui partie de la distribution *Mandriva Linux*[12]. Les techniques d'interaction qui viennent d'être mentionnées ainsi que celles qui le seront dans les deux prochaines sections sont d'ores et déjà présentes dans cette version et peuvent ainsi être utilisées au quotidien par de très nombreuses personnes.

3.2.2 Façades interactives

Comme je l'ai indiqué dans la section précédente, Metisse permet de dupliquer une fenêtre ou de copier une partie de fenêtre dans une autre, les copies étant aussi fonc-

[10] `http://www.pok3d.com/`
[11] `http://insitu.lri.fr/metisse/`
[12] `http://wiki.mandriva.com/en/Projects/Metisse`

tionnelles que la région originale. Cette possibilité est liée à la double redirection de l'affichage et des événements des périphériques d'entrée : lorsque l'utilisateur clique sur une copie, le compositeur envoie l'événement correspondant à la fenêtre originale. La copie n'existe que pour le compositeur, l'application ne connaît pas son existence. La double redirection permet également la création de trous dans les fenêtres : lorsque l'utilisateur clique dans un trou, l'événement correspondant n'est pas envoyé à la fenêtre mais à celle immédiatement en dessous, s'il en existe une.

L'idée de supprimer certaines parties des fenêtres avait été suggéré dans l'article sur Ametista [**Roussel, 2003**], ce prédécesseur de Metisse permettant déjà de modifier la forme des fenêtres (Fig. 3.4, page. 40). L'idée de dupliquer des fenêtres ou parties de fenêtres avait été proposée auparavant [Hutchings et Stasko, 2003, Tan *et al.*, 2004, Hutchings et Stasko, 2005]. Elle n'avait toutefois jamais été mise en œuvre de manière utilisable dans un réel système et chaque copie de région entraînait la création d'une nouvelle fenêtre. En permettant de coller les régions copiées dans des fenêtres existantes Metisse allait un pas plus loin. Un second pas fut franchi en proposant que plusieurs régions copiées puissent être placées dans une même fenêtre spécialement créée à cet effet. Les *façades interactives* étaient nées.

L'expression *façades interactives* fait référence à un ensemble de techniques basées sur la redirection d'entrées/sorties permettant la personnalisation d'applications existantes sans qu'il soit nécessaire de les modifier [**Stuerzlinger *et al.*, 2006**]. Plusieurs principes ont guidé la conception de ces techniques. Tout d'abord, nous voulions que la personnalisation soit simple, rapide, qu'elle ne nécessite pas d'être planifiée mais puisse se faire dans l'instant, au moment voulu. Ensuite, nous voulions que la portée spatiale et temporelle des modifications puisse être contrôlée. Il devait être ainsi possible de ne modifier l'interaction avec une application que pour un document particulier. Nous souhaitions aussi que le choix des éléments de personnalisation ne se limite pas à un ensemble prédéfini mais que l'utilisateur puisse apporter ses propres éléments. Nous voulions enfin permettre la combinaison ou l'association de différentes applications.

Les Façades[13] permettent la création de trous et la duplication de fenêtres. Elles permettent surtout la copie de portions d'interface d'une fenêtre à une autre par une interaction simple et directe. A l'aide de la souris et d'une touche spéciale du clavier, l'utilisateur sélectionne une ou plusieurs régions rectangulaires. Cette sélection est facilitée par un mécanisme d'attraction qui aide à suivre les contours des widgets. L'utilisateur effectue ensuite une opération de glisser-déposer. Le début de l'opération duplique les régions sélectionnées. Si ces copies sont lâchées sur le bureau, une nouvelle fenêtre est créé pour les contenir : une *façade*. Si les copies sont lâchées sur une façade existante, elles sont ajoutées aux éléments déjà présents, la façade étant automatiquement redimensionnée en cas de besoin (Fig. 3.8). Les copies lâchées sur une fenêtre "normale" viennent s'y coller, recouvrant le contenu original.

[13]Dans un esprit de concision, les expressions *les Façades*, *le système Façades* ou encore *le système* seront utilisées pour désigner l'ensemble des techniques mettant en œuvre l'idée de "façades interactives".

FIG. 3.8: Exemple de création d'une façade : le premier glisser-déposer crée la façade (images de gauche), le deuxième y ajoute un second élément (images de droite)

Le guidage de la sélection est rendu possible par l'obtention via une API d'accessibilité[14] de la description du contenu des fenêtres. La figure 3.9 illustre les différents flux d'information correspondant à une façade regroupant deux éléments venant de deux applications différentes. Une même région source peut être utilisée dans plusieurs façades

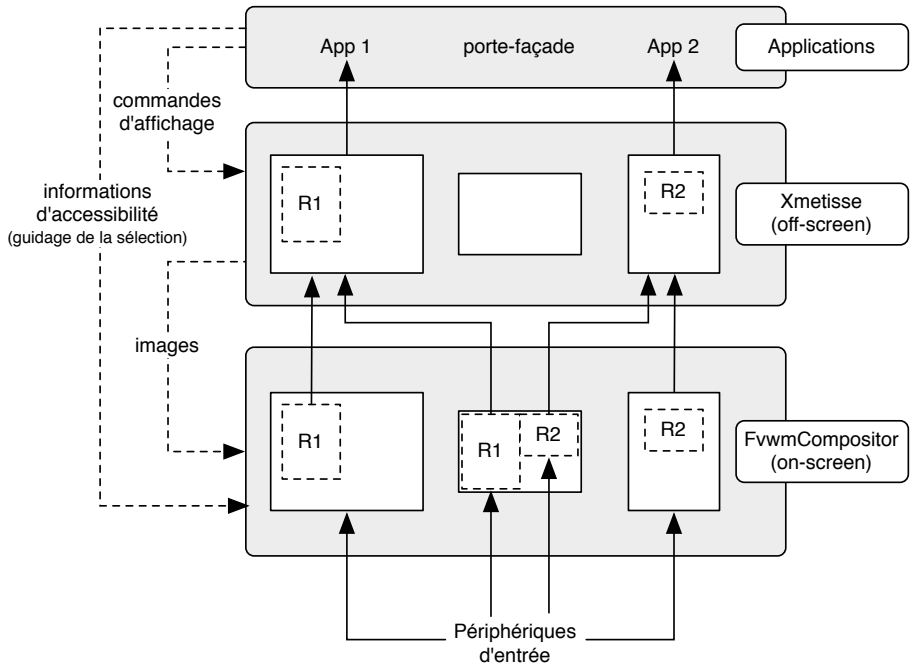


FIG. 3.9: Flux d'information associés à une façade regroupant deux éléments ($R1$ et $R2$) de deux applications différents ($App1$ et $App2$)

et une façade peut contenir un nombre arbitraire d'éléments copiés. Le système Façades assure un lien direct fonctionnel entre chaque élément copié et sa région d'origine. Il fait

[14]Les API d'accessibilité permettent à un programme d'obtenir une description de l'interface et des commandes offertes par un autre programme, d'accéder au contenu des widgets et de déclencher les actions qui y sont liées.

en sorte que les fenêtres transitoires (e.g. popup menus, tooltips) apparaissent au bon endroit, près du pointeur de la souris et non nécessairement de la région source. Il prend également les mesures appropriées en cas d'iconification, de redimensionnement ou de fermeture des fenêtres source. Une commande ajoutée au menu standard d'une façade permet à l'utilisateur de sauvegarder une description de celle-ci basée sur la géométrie, la classe, les ressources utilisées et éventuellement le nom des divers éléments. La façade peut ensuite être rappelée, manuellement ou automatiquement, lorsque les fenêtres sources sont à nouveau présentes.

La duplication de fenêtres est intéressante dans le cas de configurations multi-écrans ou d'écrans à très haute résolution dans la mesure où elle permet de placer des copies des fenêtres les plus utilisées à différents endroits, réduisant ainsi les distances à parcourir pour y accéder. La copie d'une partie de fenêtre dans une autre permet de contrôler la première application depuis la seconde. Le bouton *Reload* d'un navigateur Web peut ainsi être placé dans l'éditeur de texte avec lequel on modifie le document affiché. Les Façades peuvent également être utilisée pour créer des "télécommandes universelles" ou des "panneaux d'affichage universels", des façades regroupant quelques boutons ou zones d'affichage d'applications différentes mais utilisées dans un même contexte.

Un usage plus original des Façades consiste à remplacer une partie de l'interface d'une application par une autre interface. Ainsi, dans le cas de saisies répétées d'un formulaire comportant une liste déroulante (Fig. 3.10, image de gauche), le système est capable de remplacer cette liste par un ensemble de boutons radio ne proposant que quelques choix. La liste des choix possibles est obtenue de manière automatique par le système en interrogeant l'application à personnaliser à l'aide de l'API d'accessibilité. Une application auxiliaire est utilisée pour demander à l'utilisateur les choix qu'il souhaite conserver (Fig. 3.10, au centre). Cette application crée les boutons radio correspondants qui peuvent ensuite être copiés dans l'application à personnaliser, au-dessus de l'interface originale (Fig. 3.10, image de droite). Lorsqu'un des boutons radio est enfoncé, l'application auxiliaire sélectionne l'élément correspondant de la liste originale par l'intermédiaire de l'API d'accessibilité.

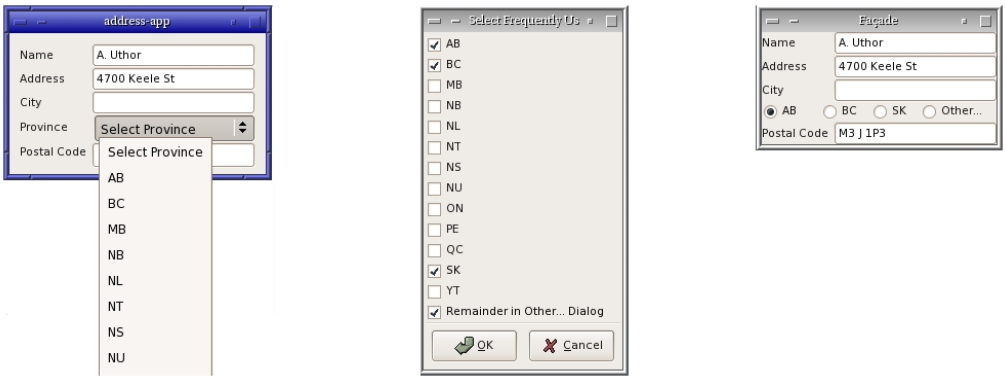


FIG. 3.10: Exemple de substitution d'interface : application originale, dialogue permettant le choix des éléments de la liste à conserver et application personnalisée

Un usage plus intéressant encore des Façades consiste à associer une façade précédemment créée à un dispositif de pointage pour la transformer en *toolglass*[15] [Bier *et al.*, 1993]. Une fois cette association activée, la façade devient semi-transparente. Elle reste affichée au-dessus des autres fenêtres et sa position est contrôlé par le dispositif qui lui est associé (e.g. un *trackball*). Lorsqu'un bouton de la souris est enfoncé au-dessus de la façade, un clic lui est d'abord envoyé pour activer l'outil choisi, puis l'événement et ceux qui suivent sont envoyés à la fenêtre qui se trouve en dessous afin de l'utiliser. Cet exemple illustre la puissance du couple Façades+Metisse qui permet pour la première fois d'interagir avec des applications existantes à l'aide d'une technique bimanuelle dont on sait depuis plus de dix ans qu'elle est plus efficace que l'interaction classique unimanuelle [Kabbash *et al.*, 1994].

L'exemple précédent tire avantageusement parti de la redirection d'entrée permise par FvwmCompositor mais comporte toujours une partie visuelle. Le concept de façade peut également s'appliquer au seul cas de la redirection d'entrée. En utilisant l'API d'accessibilité pour trouver les commandes permettant de faire défiler un document et de changer son niveau de zoom, les Façades proposent ainsi automatiquement l'utilisation de la technique OrthoZoom[16] [Appert et Fekete, 2006] dans les fenêtres où elle est possible, toujours sans nécessiter la moindre modification des applications. Au-delà de la personnalisation d'interfaces, cet exemple montre une nouvelle fois l'intérêt des Façades pour intégrer simplement dans un réel système de fenêtrage des techniques d'interaction avancées.

[15]Une toolglass est une palette d'outils flottante destinée à un usage bimanuel. La main non-dominante contrôle la position de la palette. La main dominante est utilisée pour choisir un de ses outils et l'appliquer immédiatement à travers elle sur les objets situés en dessous.

[16]OrthoZoom est une technique de pointage multi-échelles 1D qui utilise un périphérique de pointage 2D standard (e.g. une souris) pour contrôler le facteur de zoom selon la dimension orthogonale à celle utilisée pour la navigation.

3.2.3 Rock and roll !

Après avoir vu comment Metisse permet, à travers les Façades, la copie de régions interactives de l'écran, nous allons maintenant voir qu'il peut aussi faciliter la copie de données entre des documents affichés dans différentes fenêtres.

Le copier-coller est la technique de base utilisée pour répliquer une partie de document dans un autre. Disponible dès les premiers systèmes interactifs tels que Sketchpad [Sutherland, 1963] ou NLS [Engelbart, 1984], c'est l'un des services fondamentaux offerts par tous les systèmes interactifs actuels. Au fil des années, plusieurs techniques se sont imposées pour réaliser cette opération, comme l'utilisation des raccourcis clavier Ctrl-C et Ctrl-V. Mais bien que ces techniques soient utilisées chaque jour par des millions de personnes, elles restent mal comprises et sont souvent implémentées de manière différente entre les systèmes et parfois même entre les applications. Indépendamment de la technique utilisée, le copier-coller entre deux fenêtres impose souvent la manipulation de celles-ci. Lorsqu'elles se recouvrent, par exemple, l'utilisateur est parfois obligé de les déplacer ou de changer leur place dans l'ordre d'affichage (Fig. 3.11).

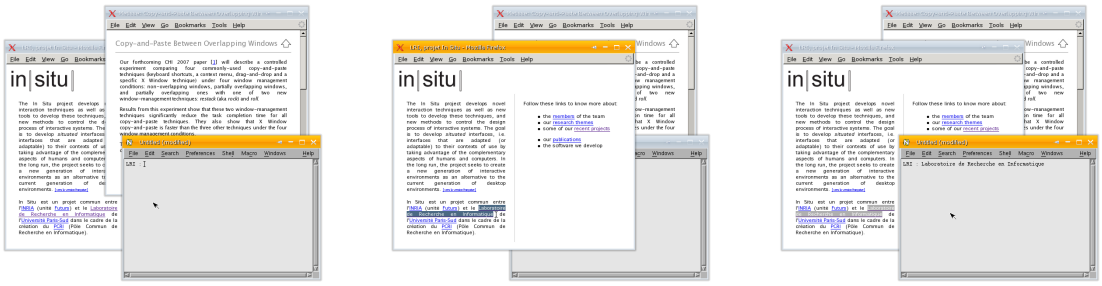


FIG. 3.11: Changement de l'ordre d'affichage lors d'une opération de copier-coller : le copier-coller de texte depuis une fenêtre initialement en arrière-plan (image de gauche) va placer celle-ci au premier plan (image du milieu) lors de la copie, puis au second lors du collage (image de droite)

La superposition des fenêtres permet le multiplexage temporel et spatial de l'espace de travail en basculant d'une fenêtre à l'autre ou en les disposant côte à côte [Lampson, 1986]. Mais à mesure que le nombre de fenêtres augmente, l'accès aux fenêtres en totalité ou partiellement recouvertes devient de plus en plus difficiles. Les écrans à haute résolution ou configurations à écrans multiples ne règlent pas vraiment ce problème. Non seulement les utilisateurs de ces systèmes ouvrent plus de fenêtres [Robertson *et al.*, 2004], mais surtout, la difficulté d'interaction sur ces grandes surfaces fait qu'il est souvent plus simple de conserver les fenêtres d'intérêt proches les unes des autres. Ainsi, si les fenêtres plein-écran auront peut-être tendance à disparaître, le recouvrement partiel ou total de fenêtres continuera sans doute d'exister.

Après le travail sur les Façades, Olivier Chapuis proposa de s'intéresser aux in-

teractions entre les opérations de copier-coller et la gestion de fenêtres. Plusieurs travaux s'étaient déjà intéressés à la manière dont on pouvait accéder aux fenêtres recouvertes [Beaudouin-Lafon, 2001, Dragicevic, 2004], rendre transparentes certaines parties des fenêtres couvrantes [Ishak et Feiner, 2004] ou limiter les recouvrements [Bell et Feiner, 2000]. Metisse offrait aussi certaines techniques pour regarder temporairement derrière des fenêtres (p. 43). Le copier-coller avait été décrit comme un support à l'innovation et à la créativité [Smith *et al.*, 1982, Shneiderman, 2000]. Il avait été étudié dans des domaines spécifiques, comme les environnements de programmation [Wallace *et al.*, 2001], les éditeurs graphiques [Citrin *et al.*, 1994] ou les systèmes pervasifs [Rekimoto, 1997]. Mais la plupart des travaux visaient surtout à l'améliorer en essayant de comprendre la nature des données copiées [Wallace *et al.*, 2001, Miller et Myers, 2002, Stylos *et al.*, 2004, Bier *et al.*, 2006]. Aucune étude comparant les techniques courantes de copier-coller n'avait été réalisée, et les interactions avec la gestion de fenêtres avaient été largement ignorées.

Une étude des pratiques courantes de copier-coller réalisée auprès de 21 personnes permit d'identifier quatre principales techniques d'interaction [**Chapuis et Roussel, 2007**] : l'utilisation de raccourcis clavier (KEY), l'utilisation d'un menu contextuel (MENU), l'utilisation du glisser-déposer (DND) et une technique spécifique à X Window permettant de coller un objet sélectionné d'un simple clic sur le bouton du milieu de la souris (X). La méthode de sélection des objets à copier est identique dans ces quatre cas, mais les opérations nécessaires au déclenchement de la copie et du collage sont différentes (Fig. 3.12). L'étude montra une utilisation courante de fenêtres partiellement recouvertes. Elle confirma également certains problèmes précédemment identifiés propres aux méthodes DND et X (e.g. incertitude sur le résultat d'un glisser-déposer, l'opération pouvant copier ou déplacer, et volatilité de la sélection primaire de X Window).

Comme l'illustre la figure 3.11, les manipulations de fenêtres potentiellement imposées par un copier-coller sont liées au fait que la moindre interaction avec une fenêtre en arrière plan la fait passer au premier plan. Les systèmes de fenêtrage actuels ne fournissent quasiment aucun moyen d'indiquer un intérêt secondaire ou temporaire pour une fenêtre. [**Chapuis et Roussel, 2007**] propose une solution simple à ce problème :

> Un clic avec le bouton gauche de la souris dans une fenêtre secondaire devrait rendre cette fenêtre primaire, i.e. la rendre active en la plaçant au premier plan et en lui donnant le focus clavier. Toute autre interaction avec une fenêtre secondaire devrait être considérée comme une manifestation temporaire d'intérêt et traitée d'une manière spécifique et appropriée.

Deux nouvelles techniques d'interaction basées sur ce principe sont également proposées afin de faciliter le copier-coller entre des fenêtres se recouvrant partiellement. La première, RESTACK[17], change temporairement l'ordre d'affichage lorsqu'une sélection est initiée dans une fenêtre secondaire puis rétablit l'ordre initial une fois la sélection ter-

[17] Le nom ROCK, que j'avais proposé, ne fut finalement pas retenu... La définition donnée par le New Oxford American Dictionary décrit pourtant plutôt bien la technique proposé : "*to move gently to and fro or from side to side*".

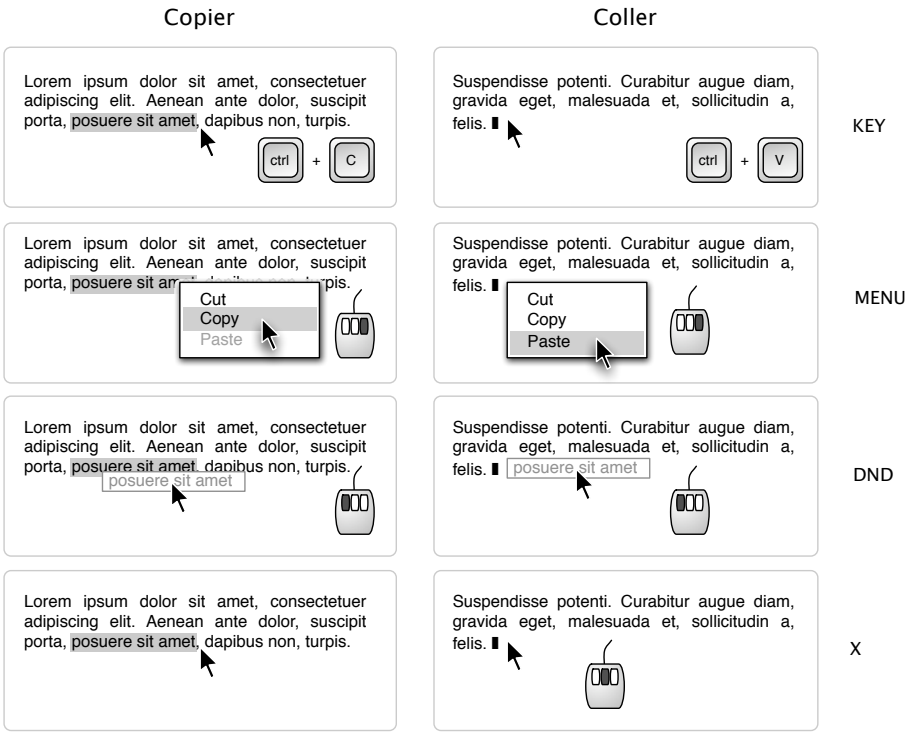


FIG. 3.12: Opérations utilisée dans les quatre principales techniques de copier-coller

minée. La seconde, ROLL, enroule sur elles-mêmes les fenêtres recouvrant celles où la sélection est initiée plus les déroule une fois celle-ci terminée (Fig. 3.13).

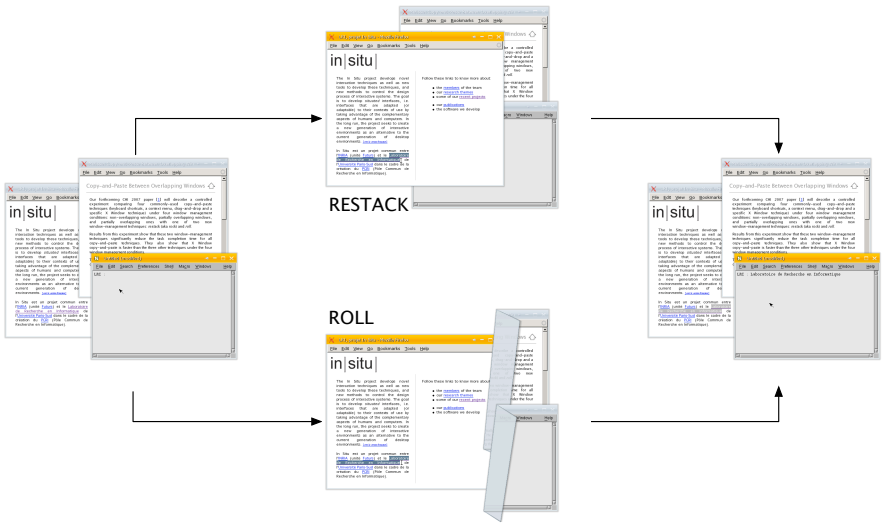


FIG. 3.13: RESTACK et ROLL

Une expérimentation fut réalisée avec 18 participants afin de comparer les quatre prin-

cipales techniques de copier-coller et d'évaluer l'impact des deux nouvelles techniques proposées. Les techniques KEY, MENU, DND et X furent testées dans quatre conditions : dans le cas de fenêtres sans recouvrement (NONOVERLAPPING), dans le cas d'un recouvrement partiel sans technique d'interaction particulière (OVERLAPPING) et dans cette situation avec l'une des deux nouvelles techniques (RESTACK et ROLL)[18]. Les résultats de cette expérimentation montrent que la technique particulière du système X Window est plus rapide que l'utilisation de raccourcis clavier, de menus contextuels ou du glisser-déposer. Ils montrent également que les techniques RESTACK et ROLL améliorent de manière significative les quatre techniques de copier-coller et suggèrent une préférence des utilisateurs pour la première.

Les techniques RESTACK et ROLL sont utilisées quotidiennement par plusieurs membres d'In Situ. Comme les Façades, elles sont d'ores et déjà disponibles dans la version de Metisse distribuée par Mandriva. Plusieurs détournements de ces techniques ont déjà été observés, comme l'utilisation de sélections arbitraires dans des fenêtres partiellement recouvertes pour en révéler temporairement le contenu. Cet usage peut être vu comme le pendant de l'opération de pliage proposée par [Beaudouin-Lafon, 2001] et implémentée par Metisse qui permettait de regarder derrière une fenêtre.

Bien qu'elle soit plus efficace, la technique X pose quelques problèmes liés à la gestion de la sélection par le système X Window. L'article [**Chapuis et Roussel, 2007**] suggère quelques pistes qui permettraient d'adapter la technique au cas du couper-coller et de résoudre ces problèmes en utilisant notamment un historique des sélections effectuées. L'intérêt de la notion d'historique de l'interaction est au cœur de la section suivante, dans laquelle sera également présenté un outil en rapport avec le copier-coller.

3.3 Vers une mémoire épisodique informatique

De juillet 2003 à juillet 2006, j'ai collaboré à Micromégas[19], un projet que j'avais co-initié avec Yves Guiard et soutenu par l'ACI Masses de Données. Ce projet portait sur l'étude de nouvelles techniques d'interaction pour la gestion de masses de données familières : des données personnelles ou professionnelles créées ou organisées par leur utilisateur, mais dont la masse est telle que leur exploitation est de plus en plus difficile.

Comme d'autres [Furnas et Rauch, 1998, Hibino, 1999, Card, 2002], nous pensions que les processus de recherche de données familières ne peuvent être étudiés qu'en les replaçant dans le contexte plus général de la gestion de ces données incluant la manière dont elles ont été produites ou obtenues, puis éventuellement classifiées et manipulées. Une première hypothèse fondamentale au projet était que l'optimisation

[18] Tous les détails de l'expérimentation (hypothèse, plan d'expérience et résultats) sont décrits dans [**Chapuis et Roussel, 2007**]

[19] Les partenaires étaient l'équipe *Pointage dans les mondes d'information* du Laboratoire Mouvement et Perception, les projets In Situ et MErLIn de l'INRIA et le Pôle Informatique de l'Institut Pasteur. Pour plus de détails sur ce projet, consulter `http://insitu.lri.fr/micromegas/`

de l'interaction entre l'utilisateur et ses données passe par un équilibre entre recherche automatique et navigation active, une hypothèse corroborée par des travaux récents [Ravasio *et al.*, 2004, Teevan *et al.*, 2004]. Une seconde hypothèse était qu'une représentation multi-échelles des données permettrait de les organiser et de les retrouver de manière plus efficace qu'avec les techniques actuelles.

Outre les travaux autour de Metisse qui s'inscrivaient dans ce projet par leur dimension quotidienne, je me suis plus personnellement intéressé à la capture d'informations décrivant le contexte d'utilisation d'une donnée particulière (e.g. une photo, un courrier électronique ou une page Web) et à l'utilisation potentielle de ces informations pour retrouver la donnée ou recréer le contexte en question.

3.3.1 Mémoire épisodique et interaction homme-machine

Une partie du travail réalisé dans le cadre de Micromégas visait à mieux comprendre les processus cognitifs d'organisation, de mémorisation et de recherche des données familières. Une étude exploratoire fut ainsi conduite à l'INRIA Rocquencourt sous la forme d'entretiens semi-dirigés et d'une quasi-expérience afin de déterminer les attributs dont les utilisateurs se souviennent concernant leurs documents papiers et électroniques [Blanc-Brude et Scapin, 2007]. Plusieurs séries d'ateliers participatifs furent également organisées à l'Institut Pasteur afin de mieux comprendre comment les biologistes retrouvent les données et outils d'analyse qu'ils utilisent et les divers moyens qu'ils mettent en œuvre pour faciliter leurs recherches.

Dès le démarrage du projet, nous nous sommes également intéressés à la capture automatique d'informations qualitatives et quantitatives concernant l'activité informatique quotidienne d'une personne : courriers reçus et envoyés, fichiers manipulés, documents imprimés, pages Web consultées, applications utilisées, organisation des fenêtres à l'écran et des icônes sur le bureau, etc. Le but initialement visé par ce travail était de générer des descriptions détaillées de l'activité pouvant être par la suite analysées afin d'en extraire des pistes utiles pour la conception de nouveaux systèmes. Il s'inscrivait ainsi dans une démarche similaire à celle d'autres chercheurs, plus axée sur la gestion de multiples tâches en parallèle [Czerwinski *et al.*, 2004, Dragunov *et al.*, 2005]. À mesure que nous progressions dans la mise en œuvre d'observateurs/enregistreurs logiciels, il apparut toutefois évident que les informations collectées pouvaient également être utiles aux personnes observées.

L'une des inspirations à l'origine du projet Micromégas était liée à l'évolution observée des techniques de recherche. Une première génération d'outils avait permis la recherche de fichiers (e.g. la commande UNIX $find$). Une seconde génération s'était intéressée au contenu des fichiers (e.g. la commande UNIX *grep* ou, de manière plus élaborée, Apple Spotlight). La troisième génération devait nécessairement s'intéresser à ce qui les entoure, au contexte. En 1998, les créateurs de Google faisaient remarquer que le texte porteur d'un lien hypertexte constitue bien souvent une description plus juste du document

pointé que le document lui-même [Brin et Page, 1998]. Là où l'algorithme de Google se révèle extrêmement puissant (et dangereux), c'est que le texte en question peut être sans rapport aucun avec le document pointé : du moment qu'un nombre suffisant d'éléments établissent un lien entre le texte et le document pointé, le lien est considéré valide.

Dans le cas qui nous intéressait, la question était de savoir quels pouvaient être les éléments extérieurs susceptibles d'être liés à des données familières recherchées par l'utilisateur. Mais le caractère familier de ces données fournissait une réponse. Ces données ayant été produites, reçues ou collectées par l'utilisateur, une histoire les reliait. Et ce passé commun avait de grande chances d'être présent à l'esprit de l'utilisateur au moment de la recherche. La mémoire épisodique permet à un sujet de se rappeler du contexte particulier d'un événement qu'il a personnellement vécu. Notre travail sur l'enregistrement automatique de l'activité fut basé sur l'idée qu'un historique détaillé de l'interaction fournirait un contexte similaire au système informatique et permettrait ainsi à l'utilisateur de faire appel à sa propre mémoire épisodique – et non sémantique – pour retrouver des données.

La notion d'historique est très faiblement présente dans les environnements interactifs que nous utilisons quotidiennement. Si certaines applications mettent en œuvre des mécanismes permettant d'annuler ou d'exécuter à nouveau une commande, rares sont celles qui se souviennent des actions effectuées sur un document une fois celui-ci refermé. Bien qu'il soit souvent possible de lister les n documents ou applications les plus récemment utilisés, ces listes sont généralement limitées en taille et ne donnent qu'une indication imprécise d'ordre relatif. Ce que les navigateurs Web appellent historique se limite généralement aux dates de première et dernière visite de chaque site[20]. De même, bien que certains systèmes aient été conçus pour enregistrer pour chaque fichier la date du dernier accès, les seules informations effectivement disponibles sont généralement la date de création et la date de dernière modification[21]. L'accès à un historique complet et détaillé de l'activité informatique d'un utilisateur est donc bien plus compliqué qu'on ne le croit. La constitution de cet historique passe nécessairement par le développement d'outils logiciels spécifiques.

Un certain nombre d'outils ont été développés dans le cadre de Micromégas pour observer et enregistrer l'activité d'un utilisateur sur les plate-formes Linux et OS X [**Roussel *et al.*, 2005**, **Roussel *et al.*, 2006b**]. En introduisant des modifications au sein du noyau de ces deux systèmes, il est par exemple possible d'intercepter les opérations d'ouverture, lecture, écriture et fermeture de fichier. Tous les accès aux fichiers, y compris en lecture seule, peuvent donc être enregistrés. En modifiant le code de Mozilla ou de ses dérivés (e.g. Firefox ou Camino), il est possible de tenir à jour un historique complet des pages Web consultées même lorsqu'elles ne sont pas rechargées, quand l'utilisateur alterne entre des documents affichés dans des onglets ou fenêtres distincts par

[20] La présentation de l'historique a aussi son importance. Ainsi, celui de Firefox regroupe bien les pages visitées selon un critère temporel (*aujourd'hui*, *hier*, *il y a 2 jours*, etc.), mais le contenu de chaque groupe est ensuite trié par ordre alphabétique, ce qui le rend relativement inutile (v. 2.0.0.6).

[21] Sur certains systèmes, le simple fait de consulter la date de dernier accès à un fichier modifie celle- ci et la rend donc inutilisable en pratique.

exemple. Des modifications similaires permettent de garder une trace des téléchargements effectués par le navigateur. L'impression de documents et l'utilisation d'outils de communication (e.g. messagerie instantanée) peuvent être déduits de l'observation de l'activité réseau liée à certains ports TCP ou UDP.

En liaison avec les travaux autour de Metisse, nous avons également développé plusieurs outils d'observation du système de fenêtrage [**Roussel *et al.*, 2005**, Chapuis, 2005]. La création, la destruction, le redimensionnement, le déplacement ou le changement d'état des fenêtres peuvent ainsi être observés, ainsi que les opérations mettant en jeu plusieurs fenêtres (e.g. copier/coller ou glisser/déposer). Le titre des fenêtres permet dans certains cas de faire le lien avec l'activité observée au niveau du système de fichiers (certaines applications incluent le nom du fichier manipulé dans le titre des fenêtres). Le nombre, la topologie et les noms des éventuels bureaux virtuels sont également accessibles. Le nom des bureaux est particulièrement intéressant puisque ceux-ci sont souvent utilisés pour séparer différentes activités.

L'historique produit par nos différents observateurs consiste en une série de flux continus d'événements. Au cours du projet, plusieurs outils de visualisation interactive ont été développés à l'aide de la boîte à outils InfoVis [Fekete, 2004] pour faciliter l'analyse exploratoire de ces flux (Fig. 3.14). Afin de pouvoir les interroger, nous nous sommes également intéressés à l'extension d'une algèbre initialement proposée par Mackay & Beaudouin-Lafon [1998] pour l'analyse de séquences vidéo. Aux opérations existantes (e.g. filtrage temporel, insertion/suppression d'événements, fusion de flux), nous avons ajouté des mécanismes d'agrégation permettant de travailler sur l'historique à différentes échelles temporelles afin de pouvoir répondre rapidement à des questions du type "quelles sont les quatre pages Web que je regarde le plus souvent en ce moment ?" ou "quels sont les articles que j'ai consultés lorsque j'ai rédigé celui-ci ?" [Shah, 2005]. D'autres travaux sont actuellement en cours au sein de l'équipe In Situ sur ces sujets.

FIG. 3.14: Exemple de visualisation interactive des flux d'événements enregistrés sur une période d'un mois

Les observateurs d'activité que nous avons développés contribuent à créer ce que l'on pourrait qualifier de *mémoire à long terme* de l'interaction, par opposition aux systèmes actuels qui ne se souviennent que de quelques événements les plus récents. La mise en œuvre de ces observateurs a toutefois soulevé de nombreux problèmes auxquels il n'a pas toujours été aisé de répondre : Quel niveau de détail choisir pour les observations ? À quel niveau se placer lorsque plusieurs points d'observation sont possibles (e.g. système d'exploitation, application ou système de fenêtrage) ? Comment stocker les flux continus d'observations, comment y accéder efficacement ? Comment éliminer le bruit éventuel, comme l'activité du système de fichier indépendante de l'utilisateur ou liée à l'observation elle-même ? L'expérience acquise à travers le projet montre que les réponses à ces questions dépendent principalement de l'usage qui est fait des enregistrements : les outils d'observation ne peuvent être conçus de manière générique mais doivent être pensés dans un but précis et spécifique, pour un usage particulier. Notre expérience montre également que les données enregistrées ne suffisent pas à déterminer leurs usages potentiels : les observations de vrais utilisateurs et les ateliers de conception participative constituent de bien meilleures sources d'inspiration [**Roussel *et al.*, 2006b**].

3.3.2 PageLinker

Une grande partie du travail réalisé par les biologistes ayant collaboré au projet Micromégas se passe sur le Web. Ils l'utilisent à la fois comme une gigantesque base de données et comme un outil d'analyse de ces données. Ils y répètent régulièrement les mêmes séries de tâches, revisitant souvent les mêmes pages, naviguant séquentiellement et en parallèle au fil de leurs hypothèses. Mais les sites Web qu'ils fréquentent et les navigateurs qu'ils utilisent ne sont pas réellement adaptés à cette pratique. Trouver les données dont ils ont besoin nécessite souvent de longues navigations, et ces données, une fois trouvées, ne pointent que rarement vers les outils d'analyse qu'ils souhaitent utiliser. Dans leur cas, comme pour de nombreuses autres personnes, les mécanismes de navigations sont insuffisants.

Malgré l'explosion des usages du Web depuis son origine, les mécanismes de navigation proposés aux utilisateurs ont très peu évolué. Et bien que des études montrent que plus de la moitié des pages vues sont en fait revisitées [Catledge et Pitkow, 1995, Tauscher et Greenberg, 1997, Cockburn et McKenzie, 2001], les mécanismes tels que les favoris ou l'historique destinés à faciliter ces revisites ne sont que rarement utilisés [Catledge et Pitkow, 1995, Tauscher et Greenberg, 1997, Weinreich *et al.*, 2006]. Concernant plus spécifiquement les biologistes, des outils existent pour automatiser certaines de leurs procédures de travail, mais ceux-ci sont également très peu utilisés. Ils sont jugés trop complexes, pas assez robustes et ne permettant pas de contrôler assez finement le déroulement des procédures. Les principaux outils utilisés par les biologistes pour retrouver un chemin parcouru sont en fait les moteurs de recherche, le courrier électronique et les Post-it.

Le besoin d'un outil pouvant servir de support à la définition de protocoles interactifs sur

le Web ressortait assez nettement des ateliers réalisés. Il fut donc décidé de développer une extension permettant de suivre le chemin parcouru par un utilisateur sur le Web afin de faciliter la navigation. Firefox fut choisi comme plate-forme de développement et il fut décidé de s'intéresser dans un premier temps aux enchaînements de pages visant à remplir un formulaire en vue d'une analyse de données [Tabard, 2005]. Afin de ne pas changer les habitudes de travail des biologistes, un moyen non intrusif fut choisi pour établir des liens entre les pages contenant les données et celles correspondant aux outils d'analyse : le navigateur fut instrumenté pour détecter les opérations de copier-coller entre deux pages, chaque opération créant un lien entre la page source et la page cible.

Les premiers essais furent assez concluants. L'extension fut rapidement détournée par les utilisateurs pour créer des liens entre des pages arbitraires, sortant ainsi du scénario initial d'analyse de données et illustrant le potentiel de l'extension comme support à une navigation contextualisée. Les utilisateurs firent toutefois plusieurs critiques, se plaignant notamment du fait que les liens n'étaient qu'unidirectionnels et qu'il était impossible de les supprimer. Un processus participatif fut donc engagé avec les biologistes afin d'améliorer la conception de l'extension. Le résultat de ce processus est PageLinker[22], une extension Firefox qui permet de créer de manière implicite ou explicite des liens entre des pages Web et d'accéder à ces liens à travers le panneau de favoris du navigateur (Fig. 3.15), mettant ainsi en œuvre les deux idées complémentaires d'historique contextuel (i.e. *vers quels autres sites ai-je l'habitude d'aller à partir de ce site ?*) et d'historique inverse (i.e. *comment suis-je arrivé à ce document ?*).

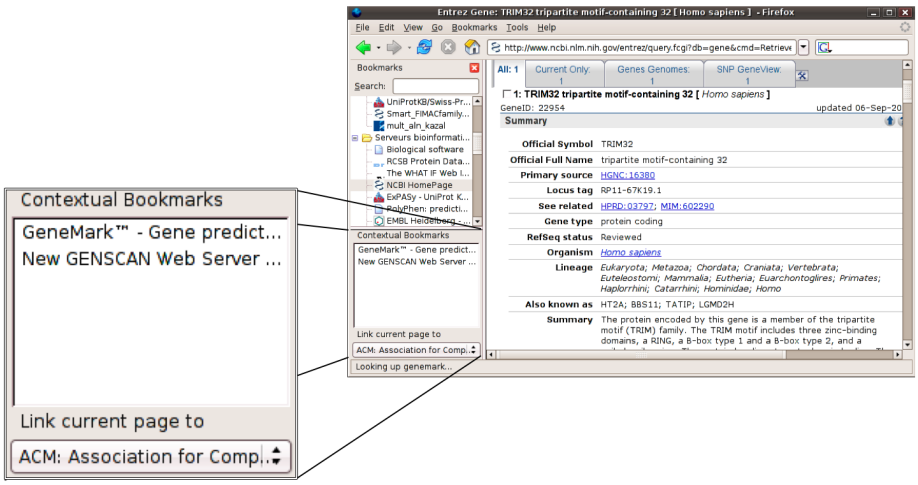


FIG. 3.15: PageLinker

Une évaluation quantitative et qualitative de PageLinker fut réalisé sous forme de quasi-expérience auprès de 12 biologistes ou bioinformaticiens [**Tabard *et al.*, 2007**]. Les données quantitatives furent recueillies à l'aide de NavTracer[23], l'un des observateurs d'activité précédemment développés dans le cadre de Micromégas [**Roussel *et al.*, 2006b**].

[22] `http://rabidfox.mozdev.org/`
[23] `http://navtracer.mozdev.org/`

Les résultats de cette évaluation montrent que l'utilisation de PageLinker réduit à la fois le temps (-28%), le nombre de clics (-22%) et le nombre de chargement de pages (-38%) nécessaires pour accomplir un scénario. Trois mois après cette évaluation, deux tiers des biologistes utilisaient encore l'extension et considéraient que les liens étaient faciles à créer (4.33/5), faciles à utiliser (4.44/5) et utiles (3.56/5).

Chapitre

4 Conclusions et perspectives

4.1 Pour la communication médiatisée

Le concept de communication multi-échelles proposé dans [**Roussel et Gueddana, 2007**] et décrit dans le chapitre 2 ouvre un large champ de recherches possibles. Un premier axe concerne la conception de services de communication correspondant à des degrés divers d'engagement, i.e. d'exposition et de sollicitation potentielle. Comme je l'ai montré à travers plusieurs exemples, les techniques de filtrage spatial et temporel du son et de l'image offrent de nombreuses possibilités pour enrichir ou dégrader la communication qui restent sans doute en partie à explorer. D'autres dimensions pourraient être rendues variables, comme le choix du moment et du correspondant. On peut être disponible ou avoir envie de communiquer sans penser nécessairement à une personne particulière. On peut aussi regretter de ne pas communiquer plus souvent avec certaines personnes. Laisser le système trouver un correspondant disponible ou un moment opportun pourrait être vu comme une manière d'alléger la communication, de réduire l'engagement.

Le deuxième axe de recherche lié à la communication multi-échelles concerne l'étude des transitions entre degrés d'engagement. Ces transitions devront être étudiées à la fois pour un service particulier, et dans un contexte multi-services. Elles posent le problème du contrôle de l'engagement lié à un service (e.g. en faisant varier la force d'un filtre spatial sur des images) et celui de la substitution ou de la combinaison de services. L'étude de ces transitions devra s'intéresser aux techniques d'interaction permettant ce contrôle, cette substitution et cette combinaison de services. L'idée de degré d'engagement variable pourrait être une nouvelle fois appliquée dans ce contexte à l'interaction homme-machine afin de permettre à la fois des interactions implicites et explicites avec le système. Le mantra de Ben Shneiderman pour la recherche d'information me semble particulièrement pertinent pour décrire l'engagement progressif que devraient permettre les transitions dans la communication avec une ou plusieurs personnes : "*d'abord une vue d'ensemble, puis du zoom et du filtrage et enfin des détails à la demande*[1]" [Shneiderman, 1996].

[1] *overview first, zoom and filter, then details-on-demand*

La mobilité des utilisateurs impose parfois des contraintes sur le degré d'engagement qu'il peuvent ou souhaitent atteindre. Il pourrait être intéressant dans ce contexte de dissocier les notions d'attention et de solicitabilité afin de permettre, dans un train par exemple, la communication textuelle dans un sens et parlée dans l'autre. L'accès intermittent au réseau et les absences prolongées posent également d'intéressants problèmes en ce qui concerne la communication périphérique. Doit-on considérer qu'en raison de leur nature a priori secondaire, les échanges manqués ne doivent pas être reproduits ? Considérant la valeur de ces échanges pour la perception de l'activité commune, doit-on au contraire les préserver ? Dans ce cas, comment les présenter efficacement ? Les transitions entre des formes de communication synchrones et asynchrones devraient être étudiées, comme les possibilités de résumé automatique.

Les travaux d'Emmanuel Nars sur la communication de groupe à travers des *appareils de communication* ouvrent également quelques perspectives. La communication simultanée avec plusieurs groupes pose ainsi d'importants problèmes liés aux effets de bord possibles des échanges, dans le cas de l'audio par exemple, ou dans le cas de conversations textuelles croisées. L'usage d'appareils de communication de différents types par les membres d'un même groupe pose également d'intéressants problèmes de traduction d'un service à un autre et de meta-protocole de communication. Sur un plan technique, les développements logiciels réalisés dans le cadre des thèses d'Emmanuel Nars et de Sofiane Gueddana pourront être intégrés à la boîte à outils Núcleo pour y introduire les concepts de communication de groupe et de communication multi-échelles. D'autres développements pourront venir compléter cette boîte à outils pour faciliter les transitions entre services de communication en s'inspirant par exemple des efforts réalisés dans le cadre du projet Open Source Telepathy[2].

Un autre thème à développer est celui de la conception de nouveaux systèmes de communication selon une approche de type système embarqué. Les systèmes sur lesquels j'ai travaillé jusqu'à présent étaient tous basés sur des plates-formes informatiques ordinaires, à usage générique (des stations de travail SGI, des ordinateurs de type PC ou Macintosh). Si le développement s'en est trouvé facilité, ces plates-formes sont toutefois surdimensionnées pour le travail qui leur est demandé, au sens propre comme en termes de capacités et de performances informatiques. Comment passer du prototype de laboratoire assemblé de façon ad hoc à un appareil de communication simple, robuste, transportable et donc plus facile à déployer ? Des solutions devraient être recherchées pour mettre en œuvre efficacement ces systèmes sur des matériels spécifiques et intégrer les éventuels composants électroniques nécessaires à leur fonctionnement (e.g. capteurs de proximité).

L'évaluation de nouveaux systèmes de communication est un sujet vaste et complexe. Comme nous avons pu le voir dans le cadre du projet européen interLiving, le déploiement de ces systèmes est difficile et grand consommateur de ressources. Les critères d'évaluation restent difficiles à déterminer dans le cas de systèmes destinés à un usage périphérique et sur le long terme. Quels peuvent être les critères pertinents concernant

[2] `http://telepathy.freedesktop.org/wiki/`

un système de communication multi-échelles ? Comment en mesurer l'impact ? Peut-on imaginer des manières d'évaluer ces systèmes qui ne nécessitent pas leur déploiement ? Des collaborations avec des chercheurs en sciences humaines et sociales devraient permettre d'avancer sur ces sujets.

Un dernier point qui mériterait d'être étudié est la manière dont ce type de recherche peut faire l'objet de transferts vers l'industrie. Alors que les dernières avancées en matière de technologies réseau et multimedia se retrouvent très rapidement dans les outils de communication commerciaux, il me semble que les usages sont trop souvent perçus par les fournisseurs de ces outils comme un phénomène à observer et non un champ à explorer activement.

4.2 Pour l'interaction homme-machine

Comme on a pu le voir à travers les exemples du chapitre précédent, le système Metisse constitue une plate-forme remarquable pour la conception, la mise en œuvre et l'évaluation de nouvelles techniques d'interaction pour le bureau informatique. Deux évolutions majeures de ce système sont d'ores et déjà envisagées. La première consiste à permettre l'utilisation simultanée de plusieurs surfaces interactives telles que des PDAs, des tables ou tableaux interactifs ou des ordinateurs portables. L'architecture logicielle du système devra être adaptée dans cette optique et de nouvelles techniques d'interaction devront être étudiées pour pouvoir transférer ou répliquer tout ou partie des fenêtres sur les différentes surfaces. La seconde évolution, consistera à permettre une utilisation concurrent par plusieurs utilisateurs des surfaces. De nouvelles techniques devront là encore être développées pour permettre cet usage collaboratif d'applications initialement mono-utilisateur.

L'idée proposée dans [**Chapuis et Roussel, 2007**] de différencier l'interaction avec les fenêtres secondaires de celle avec la fenêtre primaire ouvre un espace de conception intéressant. Des techniques similaires à celles proposées pour le copier-coller (Restack et Roll) pourraient être envisagées pour d'autres opérations sur les fenêtres ou sur d'autres éléments d'interface. L'interaction avec les icônes placées sur le bureau, derrière les fenêtres, est un exemple de contexte qui mériterait d'être étudié. De manière plus générale, l'interaction avec les fenêtres et autres éléments du bureau pourrait grandement bénéficier d'une prise en compte du contexte. Le gestionnaire de fenêtres devrait pour cela connaître plus facilement les relations entre les fenêtres et les éléments d'interface qu'elles contiennent. L'utilisation d'APIs d'accessibilité par plusieurs travaux récents, dont les Façades, va dans ce sens. Mais au-delà, il faut sans doute revoir les protocoles de communication entre les applications et le système de fenêtrage.

Faciliter la compréhension du contexte d'interaction par des applications telles que le gestionnaire de fenêtres renvoie à la notion d'observation de l'activité évoquée dans le chapitre précédent. Cette observation contextualisée devrait permettre de mieux com-

prendre l'interaction au quotidien et pourrait ainsi fournir des éléments permettant de mieux évaluer les nouvelles techniques développées pour le bureau. La visualisation interactive de logs d'activité est un sujet qui devrait prendre une importance croissante dans ce contexte. L'historique de l'interaction pourra également être utilisé comme point de départ de nouvelles techniques pour l'interaction homme machine ou la communication médiatisée, i.e. comme des données à exploiter et non pas seulement à visualiser et interpréter. Dans le cas d'activités collaboratives, ces données pourraient par exemple servir à envoyer au groupe des informations sur l'activité et la disponibilité d'une personne.

La prise en compte de l'histoire des interactions entre un individu et ses données représente un changement de paradigme important pour l'accès à ces données. A supposer que ce type d'approche se généralise, il paraît vraisemblable que les utilisateurs continueront à utiliser un système de classement hiérarchique traditionnel pour certaines de leurs données. On peut toutefois également penser que certaines données n'auront plus besoin d'être classées, le contexte dans lequel elles ont été manipulées suffisant à les retrouver. Dans certains cas, la question ne sera donc peut-être plus "où ai-je mis ce document" mais "quand l'ai-je manipulé". Ce nouveau paradigme nécessitera sans doute de nouvelles métaphores et techniques d'interaction pour compléter celles du bureau que nous utilisons aujourd'hui. L'intégration de ces métaphores et techniques aux environnements existants me semble être un point essentiel. Plutôt que de créer de nouvelles applications spécifiques, il faudra explorer les moyens permettant d'enrichir les gestionnaires de fichier, agendas, outils de communication et autres applications que nous utilisons déjà pour simplifier notre quotidien.

Bibliographie

Note : les publications référencées en caractères gras sont celles dont je suis l'auteur ou le coauteur.

[Appert et Fekete, 2006] C. Appert et J.-D. Fekete. Orthozoom scroller : 1d multi-scale navigation. In *Proceedings of ACM CHI 2006 Conference on Human Factors and Computing Systems*, pages 21–30. ACM Press, Avril 2006.

[Apple, 2005] Apple. QuickTime and MPEG-4 : Now Featuring H.264. Technology brief, Apple, Avril 2005. `http://images.apple.com/quicktime/pdf/H264_Technology_Brief.pdf`.

[Aronson, 1971] S. Aronson. The sociology of the telephone. *International journal of comparative sociology*, 12 :153–167, 1971.

[Baker *et al.*, 2005] H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, et T. Malzbender. Understanding performance in coliseum, an immersive videoconferencing system. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(2) :190–210, 2005.

[Beaudouin-Lafon *et al.*, 2001] M. Beaudouin-Lafon, A. Druin, Å. Harvard, S. Lindquist, W. Mackay, C. Plaisant, Y. Sundblad, et B. Westerlund. interLiving deliverable 1.1, technology probes for families. Technical report, CID/NADA, KTH, Suède, Octobre 2001. 100 pages.

[Beaudouin-Lafon *et al.*, 2002a] M. Beaudouin-Lafon, B. Bederson, S. Conversy, A. Druin, B. Eiderbäck, H. Evans, H. Hansen, Å. Harvard, H. Hutchinson, S. Lindquist, W. Mackay, C. Plaisant, N. Roussel, Y. Sundblad, et B. Westerlund. interLiving deliverable 1.2 & 2.2, co-design and new technologies with family users. Technical report 174, CID/NADA, KTH, Suède, Septembre 2002. 124 pages.

[Beaudouin-Lafon *et al.*, 2002b] M. Beaudouin-Lafon, N. Roussel, J. Martin, J.-D. Gascuel, G. Buchner, et H. Lissek. Terminal et système de communication. Brevet d'invention FR2811501, INPI, Janvier 2002.

[Beaudouin-Lafon et Lassen, 2000] M. Beaudouin-Lafon et M. Lassen. The architecture and implementation of CPN2000, a post-WIMP graphical application. In *Proc. ACM Symposium on User Interface Software and Technology (UIST 2000)*, volume 2(2) of *CHI Letters*, pages 181–190, San Diego, USA, November 2000. ACM Press.

[Beaudouin-Lafon, 1997] M. Beaudouin-Lafon. Interaction instrumentale : de la manipulation directe à la réalité augmentée. In *Actes Neuvièmes journées francophones sur l'Interaction Homme Machine (IHM'97), Futuroscope*, septembre 1997.

[Beaudouin-Lafon, 2001] M. Beaudouin-Lafon. Novel interaction techniques for overlapping windows. In *Proceedings of UIST 2001*, pages 153–154. ACM Press, November 2001.

[Beaudouin-Lafon, 2007] M. Beaudouin-Lafon. 40 ans d'interaction homme-machine : points de repère et perspectives. Interstices, Avril 2007. `http://interstices.info/histoire-ihm`.

[Bell et Feiner, 2000] B. Bell et S. Feiner. Dynamic space management for user interfaces. In *Proceedings of the 13th ACM symposium on User interface software and technology (UIST 2000)*, pages 239–248, New York, NY, USA, 2000. ACM Press.

[Bell, 1969] *The Bell Laboratories Record*, 47(5), May/June 1969. `http://long-lines.net/tech-equip/Picturephone/BLR0569/picturephone.pdf`.

[**Besacier *et al.*, 2007**] G. Besacier, F. Vernier, O. Chapuis, et N. Roussel. Redirection d'applications existantes et nouvelles interactions pour des usages collaboratifs co-localisés sur une table interactive. In *Proceedings of IHM 2007, 19ème conférence francophone sur l'Interaction Homme-Machine*. ACM Press, International Conference Proceedings Series, Novembre 2007. Démonstration et article court. 4 pages, à paraître.

[Bier *et al.*, 1993] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, et T. D. DeRose. Toolglass and magic lenses : the see-through interface. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM Press.

[Bier *et al.*, 2006] E. A. Bier, E. W. Ishak, et E. Chi. Entity quick click : rapid text copying based on automatic entity extraction. In *CHI '06 : CHI '06 extended abstracts on Human factors in computing systems*, pages 562–567, New York, NY, USA, 2006. ACM Press.

[Blanc-Brude et Scapin, 2007] T. Blanc-Brude et D. L. Scapin. What do people recall about their documents ? : implications for desktop search tools. In *IUI '07 : Proceedings of the 12th international conference on Intelligent user interfaces*, pages 102–111, New York, NY, USA, 2007. ACM Press.

[Bly *et al.*, 1993] S. Bly, S. Harrison, et S. Irwin. Mediaspaces : Bringing people together in a video, audio and computing environment. *Communications of the ACM*, 36(1) :28–47, Janvier 1993.

[Brave *et al.*, 1998] S. Brave, H. Ishii, et A. Dahley. Tangible interfaces for remote collaboration and communication. In *CSCW '98 : Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 169–178. ACM Press, 1998.

[Brewer *et al.*, 2007] J. Brewer, A. Williams, et P. Dourish. A handle on what's going on : combining tangible interfaces and ambient displays for collaborative groups. In *TEI '07 : Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 3–10, New York, NY, USA, 2007. ACM Press.

[Brin et Page, 1998] S. Brin et L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7) :107–117, 1998.

[Buxton, 1995] W. Buxton. Ubiquitous Media and the Active Office. *Ubiquitous Video, Nikkei Electronics*, 3.27(632) :187–195, 1995. Published in Japanese only.

[Bérard, 1999] F. Bérard. *Vision par ordinateur pour l'interaction homme-machine fortement couplée*. Thèse de doctorat, Université Joseph Fourier, Grenoble (France), Novembre 1999. 218 pages.

[Card *et al.*, 1984] S. K. Card, M. Pavel, et J. Farrell. Window-based computer dialogues. In *Proceedings of Interact'84, the first IFIP Conference on Human-Computer Interaction*, pages 239–243. North-Holland, Septembre 1984.

[Card, 2002] S. Card. Information visualization. In J. A. Jacko et A. Sears, éditeurs, *Human Computer Interaction Handbook*. Lawrence Erlbaum Associates, 2002.

[Catledge et Pitkow, 1995] L. D. Catledge et J. E. Pitkow. Characterizing browsing strategies in the world-wide web. In *Proceedings of the Third International World-Wide Web conference on Technology, tools and applications*, pages 1065–1073, New York, NY, USA, 1995. Elsevier North-Holland, Inc.

[**Chapuis et Roussel, 2005**] O. Chapuis et 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. ACM Press, Octobre 2005.

[**Chapuis et Roussel, 2007**] O. Chapuis et N. Roussel. Copy-and-paste between overlapping windows. In *Proceedings of ACM CHI 2007 Conference on Human Factors and Computing Systems*, pages 201–210. ACM Press, Avril 2007.

[Chapuis, 2005] O. Chapuis. Gestion des fenêtres : enregistrement et visualisation de l'interaction. In *Proceedings of IHM 2005, 17ème conférence francophone sur l'Interaction Homme-Machine*, pages 255–258. ACM Press, International Conference Proceedings Series, Septembre 2005.

[Chatting *et al.*, 2006] D. J. Chatting, J. S. Galpin, et J. S. Donath. Presence and portrayal : video for casual home dialogues. In *MULTIMEDIA '06 : Proceedings of the 14th annual ACM international conference on Multimedia*, pages 395–401, New York, NY, USA, 2006. ACM Press.

[Citrin *et al.*, 1994] W. Citrin, D. Broodsky, et J. McWhirter. Style-based cut-and-paste in graphical editors. In *AVI '94 : Proceedings of the workshop on Advanced visual interfaces*, pages 105–112, New York, NY, USA, 1994. ACM Press.

[Cockburn et McKenzie, 2001] A. Cockburn et B. J. McKenzie. What do web users do ? an empirical analysis of web use. *International Journal of Human-Computer Studies*, 54(6) :903–922, 2001.

[Conversy *et al.*, 2003] S. Conversy, N. Roussel, H. Hansen, H. Evans, M. Beaudouin-Lafon, et W. Mackay. Partager les images de la vie quotidienne et familiale avec videoProbe. In *Proceedings of IHM 2003, 15ème conférence sur l'Interaction Homme-Machine*, pages 228–231. ACM Press, International Conference Proceedings Series, Novembre 2003.

[Conversy *et al.*, 2005] S. Conversy, W. Mackay, M. Beaudouin-Lafon, et N. Roussel. VideoProbe : Sharing Pictures of Everyday Life. Rapport de Recherche 1409, LRI, Université Paris-Sud, France, Avril 2005. 8 pages.

[Coutaz *et al.*, 1998] J. Coutaz, F. Bérard, E. Carraux, et J. L. Crowley. Early experience with the mediaspace CoMedi. In *Proceedings of EHCI'98, the 7th IFIP Working Conference on Engineering for Human-Computer Interaction*, pages 57–72. Kluwer, 1998.

[Crabtree *et al.*, 2002] A. Crabtree, T. Hemmings, et T. Rodden. Pattern-based support for interactive design in domestic settings. In *DIS '02 : Proceedings of the conference on Designing interactive systems*, pages 265–276, New York, NY, USA, 2002. ACM Press.

[Crowley *et al.*, 2000] J. L. Crowley, J. Coutaz, et F. Bérard. Perceptual user interfaces : things that see. *Communications of the ACM*, 43(3) :54–64, Mars 2000.

[Czerwinski *et al.*, 2004] M. Czerwinski, E. Horvitz, et S. Wilhite. A diary study of task switching and interruptions. In *CHI '04 : Proceedings of the 2004 conference on Human factors in computing systems*, pages 175–182. ACM Press, 2004.

[Daft et Lengel, 1984] R. Daft et R. Lengel. Information richness : a new approach to managerial behavior and organizational design. In L. Cummings et B. Staw, éditeurs, *Research in organizational behavior 6*, pages 191–233. JAI Press, 1984.

[Dennis et Valacich, 1999] A. R. Dennis et J. S. Valacich. Rethinking media richness : Towards a theory of media synchronicity. In *HICSS '99 : Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*, page 1017. IEEE Computer Society, 1999.

[Diaz-Marino et Greenberg, 2006] R. Diaz-Marino et S. Greenberg. CAMBIENCE : A Video-Driven Sonic Ecology for Media Spaces. In *Video Proceedings of ACM CSCW'06*. ACM, November 2006.

[Dietz et Leigh, 2001] P. Dietz et D. Leigh. Diamondtouch : a multi-user touch technology. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press.

[Dourish et Bly, 1992] P. Dourish et S. Bly. Portholes : Supporting Awareness in a Distributed Work Group. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 541–547. ACM Press, 1992.

[Dragicevic, 2004] P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proceedings of UIST'04, the 17th ACM Symposium on User Interface Software and Technology*, pages 193–196. ACM Press, 2004.

[Dragunov *et al.*, 2005] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, et J. L. Herlocker. Tasktracer : a desktop environment to support multi-tasking knowledge workers. In *IUI '05 : Proceedings of the 10th international conference on Intelligent user interfaces*, pages 75–82. ACM Press, 2005.

[Dumais *et al.*, 2003] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, et D. C. Robbins. Stuff I've Seen : A System for Personal Information Retrieval and Re-Use. In *Proceedings of SIGIR 2003*, pages 72–79. ACM Press, New York, Août 2003.

[Dumas *et al.*, 1999] C. Dumas, S. Degrande, C. Chaillou, G. Saugis, P. Plénacoste, et M.-L. Viaud. SPIN : A 3-D interface for cooperative work. *Virtual Reality Journal*, pages 15–25, 1999.

[Egido, 1988] C. Egido. Videoconferencing as a Technology to Support Group Work : A Review of its Failure. In *Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work*, pages 13–24. ACM Press, Septembre 1988.

[Engelbart, 1962] D. Engelbart. Augmenting Human Intellect : A Conceptual Framework. Summary report AFOSR-3233, Stanford Research Institute, Octobre 1962. 139 pages.

[Engelbart, 1984] D. C. Engelbart. Authorship provisions in augment. In *Proceedings of COMPCON '84*, pages 465–472. IEEE, 1984.

[Fekete, 2004] J.-D. Fekete. The infovis toolkit. In *Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis 04)*, pages 167–174, Austin, TX, Octobre 2004. IEEE Press.

[Finn *et al.*, 1997] K. Finn, A. Sellen, et S. Wilbur, éditeurs. *Video-Mediated Communication*. Lawrence Erlbaum, Avril 1997. 584 pages.

[Fish *et al.*, 1993] R. S. Fish, R. E. Kraut, R. W. Root, et R. E. Rice. Video as a technology for informal communication. *Commun. ACM*, 36(1) :48–61, 1993.

[Furnas et Rauch, 1998] G. W. Furnas et S. J. Rauch. Considerations for information environments and the navique workspace. In *DL '98 : Proceedings of the third ACM conference on Digital libraries*, pages 79–88, New York, NY, USA, 1998. ACM Press.

[Gaver *et al.*, 1992] W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter, et W. Buxton. Realizing a Video Environment : EuroPARC's RAVE System. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 27–35. ACM Press, 1992.

[Gaver *et al.*, 1999] B. Gaver, T. Dunne, et E. Pacenti. Cultural probes. *interactions*, 6(1) :21–29, 1999.

[Gaver *et al.*, 2004] W. W. Gaver, A. Boucher, S. Pennington, et B. Walker. Cultural probes and the value of uncertainty. *interactions*, 11(5) :53–56, 2004.

[Gemmell *et al.*, 2002] J. Gemmell, G. Bell, R. Lueder, S. Drucker, et C. Wong. MyLifeBits : Fullfilling the Memex vision. In *Proceedings of ACM Multimedia 2002*, pages 235–238. ACM Press, December 2002.

[Grayson et Anderson, 2002] D. Grayson et A. Anderson. Perceptions of proximity in video conferencing. In *CHI'02 extended abstracts on Human factors in computer systems*, pages 596–597. ACM Press, 2002.

[Greenhalgh et Benford, 1995] C. Greenhalgh et S. Benford. Massive : a collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3) :239–261, 1995.

[Grinter *et al.*, 2005] R. E. Grinter, W. K. Edwards, M. W. Newman, et N. Ducheneaut. The work to make a home network work. In *ECSCW'05 : Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, pages 469–488, New York, NY, USA, 2005. Springer-Verlag New York, Inc.

[Grinter *et al.*, 2006] R. E. Grinter, L. Palen, et M. Eldridge. Chatting with teenagers : Considering the place of chat technologies in teen life. *ACM Trans. Comput.-Hum. Interact.*, 13(4) :423–447, 2006.

[Gross *et al.*, 2003] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, et O. Staadt. blue-c : A spatially immersive display and 3D video portal for telepresence. *ACM Transactions on Graphics, Proceedings of SIGGRAPH 2000*, 22(3) :819–827, 2003.

[**Gueddana et Roussel, 2006**] S. Gueddana et N. Roussel. Pêle-mêle, a video communication system supporting a variable degree of engagement. In *Proceedings of ACM CSCW'06 Conference on Computer-Supported Cooperative Work*, pages 423–426. ACM Press, November 2006.

[Gutwin, 2002] C. Gutwin. Traces : Visualizing the Immediate Past to Support Group Interaction. In *Proceedings of Graphics Interface*, pages 43–50, May 2002.

[Hall, 1966] E. T. Hall. *The Hidden Dimension : Man's use of Space in Public and Private*. Doubleday, New York, 1966.

[Henderson et Card, 1986] A. Henderson et S. Card. Rooms : the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3) :211–243, 1986.

[Hewett *et al.*, 1992] T. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, et W. Verplank. Curricula for Human-Computer Interaction. Technical report, ACM SIGCHI Curriculum Development Group, 1992. 168 pages.

[Hibino, 1999] S. L. Hibino. A task-oriented view of information visualization. In *CHI '99 : CHI '99 extended abstracts on Human factors in computing systems*, pages 178–179, New York, NY, USA, 1999. ACM Press.

[Hindus, 1999] D. Hindus. The importance of homes in technology research. In *CoBuild '99 : Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, pages 199–207, London, UK, 1999. Springer-Verlag.

[Hollan et Stornetta, 1992] J. Hollan et S. Stornetta. Beyond being there. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 119–125. ACM Press, 1992.

[Hudson et Smith, 1996] S. E. Hudson et I. Smith. Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems. In *CSCW '96 : Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 248–257. ACM Press, November 1996.

[Hutchings et Stasko, 2003] D. Hutchings et J. Stasko. An Interview-based Study of Display Space Management. Technical Report 03-17, GIT-GVU, Mai 2003.

[Hutchings et Stasko, 2004a] D. Hutchings et J. Stasko. Revisiting Display Space Management : Understanding Current Practice to Inform Next-generation Design. In *Proceedings of GI 2004*, pages 127–134. Canadian Human-Computer Communications Society, Juin 2004.

[Hutchings et Stasko, 2004b] D. Hutchings et J. Stasko. Shrinking window operations for expanding display space. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 350–353. ACM Press, 2004.

[Hutchings et Stasko, 2005] D. Hutchings et J. Stasko. mudibo : Multiple dialog boxes for multiple monitors. In *Extended abstracts of ACM CHI 2005 Conference on Human factors and Computing Systems*. ACM Press, Avril 2005.

[**Hutchinson *et al.*, 2003**] H. Hutchinson, W. Mackay, B. Westerlund, B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, B. Eiderbäck, S. Lindquist, et Y. Sundblad. Technology probes : Inspiring design for and with families. In *Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems*, volume 5(1) of *CHI Letters*, pages 17–24. ACM Press, Avril 2003.

[Isaacs *et al.*, 1997] E. Isaacs, S. Whittaker, D. Frohlich, et B. O'Conaill. Informal communication re-examined : New functions for video in supporting opportunistic encounters. In K. Finn, A. Sellen, et S. Wilbur, éditeurs, *Video-mediated communication*. Lawrence Erlbaum Associates, 1997.

[Isaacs *et al.*, 2002] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, et C. Kamm. The character, functions, and styles of instant messaging in the workplace. In *CSCW '02 : Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 11–20, New York, NY, USA, 2002. ACM Press.

[Ishak et Feiner, 2004] E. W. Ishak et S. K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 189–192, New York, NY, USA, 2004. ACM Press.

[Ishii *et al.*, 1993] H. Ishii, M. Kobayashi, et J. Grudin. Integration of interpersonal space and shared workspace : ClearBoard design and experiments. *ACM Transactions on Information Systems (TOIS)*, 11(4) :349–375, Octobre 1993.

[Jain *et al.*, 2002] R. Jain, M. Reiser, N. Roussel, M. Sakauchi, et H. Towles. The next five years in telepresence. Panel Session, ACM Multimedia 2002 Workshop on Immersive Telepresence, Juan les Pins, France, December 2002.

[Johnson *et al.*, 1989] J. Johnson, T. L. Roberts, W. Verplank, D. C. Smith, C. Irby, M. Beard, et K. Mackey. The Xerox Star : a retrospective. *IEEE Computer*, 22(9) :11–29, Septembre 1989.

[Jouppi *et al.*, 2004] N. P. Jouppi, S. Iyer, S. Thomas, et A. Slayden. Bireality : mutually-immersive telepresence. In *MULTIMEDIA '04 : Proceedings of the 12th annual ACM international conference on Multimedia*, pages 860–867. ACM Press, 2004.

[Jouppi, 2002] N. P. Jouppi. First steps towards mutually-immersive mobile telepresence. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 354–363. ACM Press, 2002.

[Jul et Furnas, 1998] S. Jul et G. W. Furnas. Critical zones in desert fog : aids to multiscale navigation. In *UIST '98 : Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 97–106, New York, NY, USA, 1998. ACM Press.

[Kabbash *et al.*, 1994] P. Kabbash, W. Buxton, et A. Sellen. Two-handed input in a compound task. In *CHI '94 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 417–423, New York, NY, USA, 1994. ACM Press.

[Kantorowitz et Sudarsky, 1989] E. Kantorowitz et O. Sudarsky. The adaptable user interface. *Communications of the ACM*, 32(11) :1352–1358, 1989.

[Karahalios et Donath, 2004] K. Karahalios et J. Donath. Telemurals : linking remote spaces with social catalysts. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 615–622, New York, NY, USA, 2004. ACM Press.

[Kay, 1968] A. Kay. FLEX - A flexible extendable language. Master's thesis, University of Utah, USA, 1968.

[Kay, 1993] A. Kay. The early history of smalltalk. *SIGPLAN Not.*, 28(3) :69–95, 1993.

[Kaye, 2001] J. N. Kaye. Symbolic Olfactory Display. Master's thesis, Massachusetts Institute of Technology (USA), Mai 2001. 144 pages.

[Kurtenbach et Buxton, 1994] G. Kurtenbach et W. Buxton. User learning and performance with marking menus. In *CHI '94 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 258–264, New York, NY, USA, 1994. ACM Press.

[Kühme, 1993] T. Kühme. A user-centered approach to adaptive interfaces. In *IUI '93 : Proceedings of the 1st international conference on Intelligent user interfaces*, pages 243–245, New York, NY, USA, 1993. ACM Press.

[Lampson, 1986] B. Lampson. Personal distributed computing : the alto and ethernet software. In *Proceedings of the ACM Conference on The history of personal workstations*, pages 101–131. ACM Press, 1986.

[Lipartito, 2003] K. Lipartito. PicturePhone and the Information Age : The Social Meaning of Failure. *Technology and Culture*, 44(1) :50–81, Janvier 2003.

[Lombard et Ditton, 1997] M. Lombard et T. Ditton. At the heart of it all : The concept of presence. *Journal of Computer-Mediated Communication*, 3(2), Septembre 1997.

[Mackay *et al.*, 2003] W. Mackay, H. Evans, H. Hansen, L. Dachary, et N. Gaudron. Weaving an interactive thread : An interactive event for Tales. In *Proceedings of Tales of the Disappearing Computer*, pages 409–415, Santorini, Greece, Juin 2003.

[Mackay *et al.*, 2004] W. Mackay, M. Beaudouin-Lafon, et N. Gaudron. Dispositif de contrôle de communications. Brevet d'invention WO/2006/027424, Organisation Mondiale de la Propriété Intellectuelle, Août 2004.

[Mackay et Beaudouin-Lafon, 1998] W. Mackay et M. Beaudouin-Lafon. DIVA : Exploratory data analysis with multimedia streams. In *Proc. ACM Human Factors in Computing Systems, CHI'98, Los Angeles (USA)*, pages 416–423. ACM Press, Avril 1998.

[Mackay et Beaudouin-Lafon, 2005] W. Mackay et M. Beaudouin-Lafon. Familynet : A tangible interface for managing intimate social networks. In *Proceedings of SOUPS'05, Symposium On Usable Privacy and Security*. ACM Press, Juillet 2005.

[Mackay et Fayard, 1997] W. Mackay et A.-L. Fayard. HCI, natural science and design : a framework for triangulation across disciplines. In *DIS '97 : Proceedings of the conference on Designing interactive systems*, pages 223–234, New York, NY, USA, 1997. ACM Press.

[Mackay, 1999] W. Mackay. Media Spaces : Environments for Informal Multimedia Interaction. In M. Beaudouin-Lafon, éditeur, *Computer-Supported Co-operative Work, Trends in Software Series*. John Wiley & Sons Ltd, 1999.

[Mackay, 2004] W. Mackay. The interactive thread : Exploring methods for multidisciplinary design. In *DIS '04 : Proceedings of the 2004 conference on Designing interactive systems*, pages 103–112, New York, NY, USA, Août 2004. ACM Press.

[McGrenere *et al.*, 2002] J. McGrenere, R. M. Baecker, et K. S. Booth. An evaluation of a multiple interface design solution for bloated software. In *CHI '02 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 164–170, New York, NY, USA, 2002. ACM Press.

[Miller et Myers, 2002] R. C. Miller et B. A. Myers. Multiple selections in smart text editing. In *IUI '02 : Proceedings of the 7th international conference on Intelligent user interfaces*, pages 103–110, New York, NY, USA, 2002. ACM Press.

[Mynatt *et al.*, 2001] E. D. Mynatt, J. Rowan, S. Craighill, et A. Jacobs. Digital family portraits : supporting peace of mind for extended family members. In *CHI '01 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 333–340, New York, NY, USA, 2001. ACM Press.

[Nardi *et al.*, 2000] B. A. Nardi, S. Whittaker, et E. Bradner. Interaction and outeraction : instant messaging in action. In *CSCW '00 : Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 79–88, New York, NY, USA, 2000. ACM Press.

[Nars, 2003] E. Nars. Conception de l'infrastructure logicielle de FamilyNet. Rapport de stage de DEA, Université Paris-Sud, Orsay, France, Septembre 2003. 47 pages.

[Nars, 2004a] E. Nars. Communiquer par groupes avec Circa. In *Proceedings of IHM 2004*, pages 235–238. ACM, International Conference Proceedings Series, Août 2004.

[Nars, 2004b] E. Nars. A group communication infrastructure. In *NordiCHI '04 : Proceedings of the third Nordic conference on Human-computer interaction*, pages 429–432. ACM Press, November 2004.

[Nars, 2005] E. Nars. Communication appliances tools. In *Proceedings of SOUPS'05, Symposium On Usable Privacy and Security*. ACM Press, Juillet 2005.

[Nars, 2007] E. Nars. *Support informatique à des communications de groupe*. Thèse de doctorat, Université Paris-Sud, Orsay, France, Septembre 2007. 183 pages (version préliminaire).

[Neustaedter *et al.*, 2002] C. Neustaedter, S. Greenberg, et S. Carpendale. IMVis : instant messenger visualization. In *Video Proceedings of ACM CSCW'02*. ACM, November 2002.

[Neustaedter *et al.*, 2006] C. Neustaedter, S. Greenberg, et M. Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction*, 13(1) :1–36, 2006.

[Nguyen et Canny, 2005] D. Nguyen et J. Canny. Multiview : spatially faithful group video conferencing. In *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 799–808, New York, NY, USA, 2005. ACM Press.

[Okada *et al.*, 1994] K.-I. Okada, F. Maeda, Y. Ichikawaa, et Y. Matsushita. Multiparty videoconferencing at virtual social distance : MAJIC design. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pages 385–393. ACM Press, Octobre 1994.

[Perlin et Fox, 1993] K. Perlin et D. Fox. Pad : An alternative approach to the computer interface. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 57–64. ACM Press, 1993.

[Quan *et al.*, 2003] D. Quan, D. Huynh, et D. R. Karger. Haystack : A Platform for Authoring End User Semantic Web Applications. In *The SemanticWeb - Proceedings of ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*. Springer, Octobre 2003.

[Ravasio *et al.*, 2004] P. Ravasio, S. G. Schär, et H. Krueger. In Pursuit of Desktop Evolution : User Problems and Practices With Modern Desktop Systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 11(2) :156–180, Juin 2004.

[Rekimoto, 1997] J. Rekimoto. Pick-and-drop : a direct manipulation technique for multiple computer environments. In *UIST '97 : Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39, New York, NY, USA, 1997. ACM Press.

[Richardson *et al.*, 1998] T. Richardson, Q. Stafford-Fraser, K. R. Wood, et A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1) :33–38, Jan-Feb 1998.

[Riche et Mackay, 2007] Y. Riche et W. Mackay. Markerclock : A communicating augmented clock for elderly. In *Interact 2007 : Proceedings of the IFIP 2007 International Conference on Human-Computer Interaction*, 2007. 5 pages, à paraître.

[Riesenbach *et al.*, 1995] R. Riesenbach, W. Buxton, G. Karam, et G. Moore. Ontario Telepresence Project. Final report, Information technology research centre, Telecommunications research institute of Ontario, Mars 1995. `http://www.dgp.toronto.edu/tp/techdocs/Final_Report.pdf`.

[Robertson *et al.*, 2000] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, et V. Gorokhovsky. The Task Gallery : a 3D window manager . In *Proc. of ACM CHI 2000 Conference on Human Factors in Computing Systems*, pages 494–501. ACM Press, Avril 2000.

[Robertson *et al.*, 2004] G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. Hutchings, B. Meyers, D. Robbins, et G. Smith. Scalable Fabric : Flexible Task Management. In *Proc. of AVI 2004*, pages 85–89, Mai 2004.

[Roussel *et al.*, 2003] N. Roussel, H. Evans, et H. Hansen. Utilisation de la distance comme interface à un système de communication vidéo. In *Proceedings of IHM 2003, 15ème conférence sur l'Interaction Homme-Machine*, pages 268–271. ACM Press, International Conference Proceedings Series, Novembre 2003.

[Roussel *et al.*, 2004a] N. Roussel, H. Evans, et H. Hansen. MirrorSpace : using proximity as an interface to video-mediated communication. In A. Ferscha et F. Mattern, éditeurs, *Proceedings of Pervasive 2004, the second international conference on Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 345–350. Springer, Avril 2004.

[Roussel *et al.*, 2004b] N. Roussel, H. Evans, et H. Hansen. Proximity as an interface for video communication. *IEEE Multimedia*, 11(3) :12–16, July-September 2004.

[Roussel *et al.*, 2005] N. Roussel, J. Fekete, et M. Langet. Vers l'utilisation de la mémoire épisodique pour la gestion de données familières. In *Proceedings of IHM 2005, 17ème conférence francophone sur l'Interaction Homme-Machine*, pages 247–250. ACM Press, International Conference Proceedings Series, Septembre 2005.

[Roussel *et al.*, 2006a] N. Roussel, H. Evans, et H. Hansen. More about MirrorSpace. Rapport de Recherche 1438, LRI, Université Paris-Sud, France, Mars 2006. 8 pages.

[Roussel *et al.*, 2006b] N. Roussel, A. Tabard, et C. Letondal. All you need is log. Position paper, WWW 2006 Workshop on Logging Traces of Web Activity : The Mechanics of Data Collection, Mai 2006. 4 pages.

[Roussel et Chapuis, 2005] N. Roussel et O. Chapuis. Metisse : un système de fenêtrage hautement configurable et utilisable au quotidien. In *Proceedings of IHM 2005, 17ème conférence francophone sur l'Interaction Homme-Machine*, pages 279–282. ACM Press, International Conference Proceedings Series, Septembre 2005.

[Roussel et Gueddana, 2007] N. Roussel et S. Gueddana. Beyond "Beyond being there" : towards multiscale communication systems. In *Proceedings of ACM Multimedia 2007*. ACM Press, Septembre 2007. 9 pages, to be published.

[Roussel et Nouvel, 1999] N. Roussel et G. Nouvel. La main comme télépointeur. In *Tome 2 des actes des 11ème journées francophones sur l'Interaction Homme Machine (IHM'99)*, pages 33–36, Novembre 1999.

[Roussel, 1999] N. Roussel. Mediascape : a Web-based Mediaspace. *IEEE Multimedia*, 6(2) :64–74, April-June 1999.

[Roussel, 2000a] N. Roussel. *Support informatique à une communication médiatisée*. Thèse de doctorat, Université Paris-Sud, Orsay, France, Juillet 2000. 190 pages.

[Roussel, 2000b] N. Roussel. Web-Based Mediaspace. In B. Furht, éditeur, *Handbook of Internet Computing*, chapter 9, pages 205–226. CRC Press, Boca Raton, Florida, Juin 2000.

[Roussel, 2001] N. Roussel. Exploring new uses of video with videoSpace. In M. R. Little et L. Nigay, éditeurs, *Proceedings of EHCI'01, the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, volume 2254 of *Lecture Notes in Computer Science*, pages 73–90. Springer, 2001.

[Roussel, 2002a] N. Roussel. Experiences in the design of the well, a group communication device for teleconviviality. In *Proceedings of ACM Multimedia 2002*, pages 146–152. ACM Press, December 2002.

[Roussel, 2002b] N. Roussel. Le puits : un dispositif de communication audio-vidé pour la téléconvivialité. In *Proceedings of IHM 2002, 14ème conférence sur l'Interaction Homme-Machine*, pages 227–230. ACM Press, International Conference Proceedings Series, Novembre 2002.

[Roussel, 2002c] N. Roussel. VideoWorkspace : une boîte à outils pour l'exploration de nouvelles techniques de gestion de fenêtres. In *Proceedings of IHM 2002, 14ème conférence sur l'Interaction Homme-Machine*, pages 271–274. ACM Press, International Conference Proceedings Series, Novembre 2002.

[Roussel, 2003] N. Roussel. Ametista : a mini-toolkit for exploring new window management techniques. In *Proceedings of CLIHC 2003, the 1st Latin American Conference on Human-Computer Interaction*, pages 117–124. ACM Press, Août 2003.

[Roussel, 2006a] N. Roussel. From analog to digital, from the office to the living room : why I happily worked in a media space but don't live in one. Position paper, ACM CSCW 2006 Workshop "Media Space - Reflecting on 20 Years", November 2006. 4 pages.

[Roussel, 2006b] N. Roussel. POMDOM : Paradigmes, outils et méthodes pour l'interaction en environnement domestique. Atelier organisé dans le cadre de la conférence UbiMob'06, Septembre 2006.

[Rowe et Jain, 2005] L. A. Rowe et R. Jain. ACM SIGMM retreat report on future directions in multimedia research. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(1) :3–13, 2005.

[Saint-Andre, 2004a] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP) : Core. Standard, RFC-3920, IETF, Octobre 2004.

[Saint-Andre, 2004b] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP) : Instant Messaging and Presence. Standard, RFC-3921, IETF, Octobre 2004.

[Schmidt et Bannon, 1992] K. Schmidt et L. Bannon. Taking CSCW seriously : Supporting articulation work. *Journal of Computer Supported Cooperative Work*, 1(1-2) :7–40, Mars 1992.

[Shah, 2005] M. A. Shah. Analysis of Temporal Data. Rapport de stage de Master Recherche, Université Paris-Sud, Orsay, France, Septembre 2005. 18 pages.

[Shen *et al.*, 2004] C. Shen, F. Vernier, C. Forlines, et M. Ringel. Diamondspin : an extensible toolkit for around-the-table interaction. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 167–174, New York, NY, USA, 2004. ACM Press.

[Shneiderman, 1996] B. Shneiderman. The eyes have it : A task by data type taxonomy for information visualizations. In *VL '96 : Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society, 1996.

[Shneiderman, 2000] B. Shneiderman. Creating creativity : user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction*, 7(1) :114–138, 2000.

[Short *et al.*, 1976] J. Short, E. Williams, et B. Christie. *The Social Psychology of Telecommunications*. Wiley, New York, 1976.

[Smale et Greenberg, 2005] S. Smale et S. Greenberg. Broadcasting information via display names in instant messaging. In *GROUP'05 : Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 89–98, New York, NY, USA, 2005. ACM Press.

[Smith *et al.*, 1982] D. C. Smith, C. Irby, R. Kimball, W. Verplank, et E. Harslem. Designing the Star user interface. *Byte*, 7(4) :242–282, 1982.

[Smith et Hudson, 1995] I. Smith et S. E. Hudson. Low disturbance audio for awareness and privacy in media space applications. In *MULTIMEDIA '95 : Proceedings of the third ACM international conference on Multimedia*, pages 91–97. ACM Press, 1995.

[Strong et Gaver, 1996] R. Strong et B. Gaver. Feather, scent and shaker : supporting simple intimacy. In *Proceedings of ACM CSCW'96 Conference on Computer-Supported Cooperative Work*, pages 29–30. ACM Press, November 1996.

[**Stuerzlinger *et al.*, 2006**] W. Stuerzlinger, O. Chapuis, D. Phillips, et N. Roussel. User Interface Façades : Towards Fully Adaptable User Interfaces. In *Proceedings of UIST'06, the 19th ACM Symposium on User Interface Software and Technology*, pages 309–318. ACM Press, Octobre 2006.

[Stylos *et al.*, 2004] J. Stylos, B. A. Myers, et A. Faulring. Citrine : providing intelligent copy-and-paste. In *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 185–188, New York, NY, USA, 2004. ACM Press.

[**Sundblad *et al.*, 2004**] Y. Sundblad, M. Beaudouin-Lafon, S. Conversy, L. Dachary, B. Eiderbäck, N. Gaudron, H. Evans, H. Hansen, H. Hutchinson, S. Lindquist, W. Mackay, C. Plaisant, N. Roussel, et B. Westerlund. interLiving deliverable 1.3 & 2.3, studies of co-designed prototypes in family contexts. Technical report 231, CID/NADA, KTH, Suède, Février 2004. 157 pages.

[Sutherland, 1963] I. Sutherland. *Sketchpad, a man-machine graphical communication system*. Thèse de doctorat, Massachusetts Institute of Technology, USA, Janvier 1963.

[Suzuki et Hashimoto, 2004] K. Suzuki et S. Hashimoto. Feellight : a communication device for distant nonverbal exchange. In *ETP '04 : Proceedings of the 2004 ACM SIGMM workshop on Effective telepresence*, pages 40–44. ACM Press, 2004.

[**Tabard *et al.*, 2007**] A. Tabard, W. Mackay, N. Roussel, et C. Letondal. Pagelinker : Integrating contextual bookmarks into a browser. In *Proceedings of ACM CHI 2007 Conference on Human Factors and Computing Systems*, pages 337–346. ACM Press, Avril 2007.

[Tabard, 2005] A. Tabard. Conception d'un navigateur web spécifique pour la bio-informatique. Rapport de stage de Master, Université Pierre et Marie Curie, Paris 6, Septembre 2005. 67 pages.

[Tan *et al.*, 2004] D. Tan, B. Meyers, et M. Czerwinski. Wincuts : manipulating arbitrary window regions for more effective use of screen space. In *CHI '04 : Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1525–1528. ACM Press, 2004.

[Tanaka *et al.*, 2004] K. Tanaka, J. Hayashi, M. Inami, et S. Tachi. Twister : An immersive autostereoscopic display. In *VR '04 : Proceedings of the IEEE Virtual Reality 2004 (VR'04)*, pages 59–66, Washington, DC, USA, 2004. IEEE Computer Society.

[Tang et Rua, 1994] J. C. Tang et M. Rua. Montage : Providing Teleproximity for Distributed Groups. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, pages 37–43. ACM Press, Avril 1994.

[Tauscher et Greenberg, 1997] L. Tauscher et S. Greenberg. Revisitation patterns in world wide web navigation. In *CHI '97 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 399–406, New York, NY, USA, 1997. ACM Press.

[Teevan *et al.*, 2004] J. Teevan, C. Alvarado, M. S. Ackerman, et D. R. Karger. The Perfect Search Engine Is Not Enough : A Study of Orienteering Behavior in Directed Search. In *Proceedings of ACM CHI 2004 Conference on Human factors and Computing Systems*, pages 415–422. ACM Press, Avril 2004.

[Wallace *et al.*, 2001] G. Wallace, R. Biddle, et E. Tempero. Smarter cut-and-paste for programming text editors. In *Proceedings of AUIC '01*, pages 56–63. IEEE, 2001.

[Wauthier, 2006] V. Wauthier. Nightboard : création d'un plafond communicant pour les couples physiquement distants. Mémoire de Master Professionnel, Université Paris-Sud, Orsay, France, Septembre 2006. 26 pages.

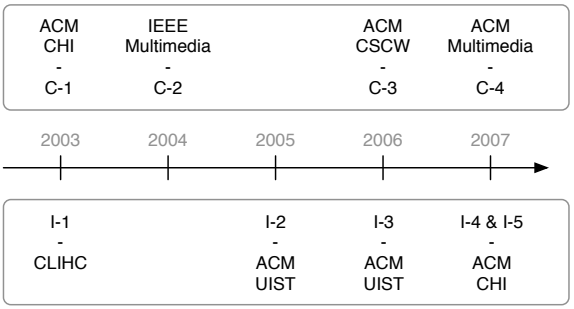
[Weinreich *et al.*, 2006] H. Weinreich, H. Obendorf, E. Herder, et M. Mayer. Off the beaten tracks : exploring three aspects of web navigation. In *WWW '06 : Proceedings of the 15th international conference on World Wide Web*, pages 133–142, New York, NY, USA, 2006. ACM Press.

[Whittaker, 1995] S. Whittaker. Video as a technology for interpersonal communications : a new perspective. In A. Rodriguez et J. Maitan, éditeurs, *Proceedings of Multimedia Computing and Networking 1995*, volume 2417, pages 294–304. SPIE, 1995.

[Zhao et Stasko, 1998] Q. A. Zhao et J. T. Stasko. Evaluating image filtering based techniques in media space applications. In *CSCW '98 : Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 11–18. ACM Press, November 1998.

Sélection d'articles de recherche

La série d'articles qui conclut ce document illustre plus en détail les principaux travaux que j'ai effectués au cours de ces dernières années. La figure ci-dessous replace les neuf articles sélectionnés dans le contexte chronologique :



Les quatre premiers articles sélectionnés portent sur la communication médiatisée :

C-1 Hilary Hutchinson, Wendy Mackay, Bosse Westerlund, Benjamin Bederson, Alison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, Nicolas Roussel, Björn Eiderbäck, Sinna Lindquist, et Yngve Sundblad. Technology probes : Inspiring design for and with families. In *Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems*, pages 17–24. ACM Press, Avril 2003.

C-2 Nicolas Roussel, Helen Evans, et Heiko Hansen. Proximity as an interface for video communication. *IEEE Multimedia*, 11(3) :12–16, July-September 2004.

C-3 Sofiane Gueddana et Nicolas Roussel. Pêle-mêle, a video communication system supporting a variable degree of engagement. In *Proceedings of ACM CSCW'06 Conference on Computer-Supported Cooperative Work*, pages 423–426. ACM Press, November 2006.

C-4 Nicolas Roussel et Sofiane Gueddana. Beyond "Beyond being there" : towards multiscale communication systems. In *Proceedings of ACM Multimedia 2007*. ACM Press, Septembre 2007. 9 pages, to be published.

Les cinq articles suivants portent sur l'interaction homme-machine :

I-1 N. Roussel. Ametista : a mini-toolkit for exploring new window management techniques. In *Proceedings of CLIHC 2003, the 1st Latin American Conference on Human-Computer Interaction*, pages 117-124, August 2003. ACM Press.

I-2 Olivier Chapuis et Nicolas 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. ACM Press, Octobre 2005.

I-3 Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, et Nicolas Roussel. User Interface Façades : Towards Fully Adaptable User Interfaces. In *Proceedings of UIST'06, the 19th ACM Symposium on User Interface Software and Technology*, pages 309–318. ACM Press, Octobre 2006.

I-4 Olivier Chapuis et Nicolas Roussel. Copy-and-paste between overlapping windows. In *Proceedings of ACM CHI 2007 Conference on Human Factors and Computing Systems*, pages 201–210. ACM Press, Avril 2007.

I-5 Aurélien Tabard, Wendy Mackay, Nicolas Roussel, et Catherine Letondal. Pagelinker : Integrating contextual bookmarks into a browser. In *Proceedings of ACM CHI 2007 Conference on Human Factors and Computing Systems*, pages 337–346. ACM Press, Avril 2007.

Technology Probes: Inspiring Design for and with Families

Hilary Hutchinson[1], Wendy Mackay[2], Bosse Westerlund[3],
[1] Benjamin B. Bederson, Allison Druin, Catherine Plaisant,
[2] Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, Nicolas Roussel,
[3] Björn Eiderbäck, Sinna Lindquist, Yngve Sundblad

[1]HCIL, UMIACS, CS
University of Maryland
College Park, MD 20742 USA
hilary@cs.umd.edu

[2]LRI, INRIA Futurs
Université de Paris-Sud
91405 Orsay Cedex, France
mackay@lri.fr

[3]CID, NADA
Kungl Tekniska Högskolan
SE-100 44 Stockholm, Sweden
bosse@nada.kth.se

ABSTRACT
We describe a new method for use in the process of co-designing technologies with users called technology probes. Technology probes are simple, flexible, adaptable technologies with three interdisciplinary goals: the social science goal of understanding the needs and desires of users in a real-world setting, the engineering goal of field-testing the technology, and the design goal of inspiring users and researchers to think about new technologies. We present the results of designing and deploying two technology probes, the messageProbe and the videoProbe, with diverse families in France, Sweden, and the U.S. We conclude with our plans for creating new technologies for and with families based on our experiences.

Keywords
Computer Mediated Communication, Home, Ethnography, Participatory Design and Cooperative Design

INTRODUCTION
In his book, *Bowling Alone* [22], Robert Putnam laments the loss of "social capital"– the interconnections we have with our family, friends, and neighbors – in American society. People participate in civic affairs less frequently, hardly know their neighbors, and socialize less often with friends. The HomeNet study at Carnegie Mellon [16, 17] indicates that computers and the Internet can contribute to this problem by isolating people from family and friends and increasing their daily stress levels.

CHI 2003, April 5–10, 2003, Ft. Lauderdale, Florida, USA.

However, this study also suggests that when used for communication, computers and the Internet can play a positive role in keeping people connected – email, instant messaging, and family web sites are just a few of the ways the Internet helps keep people in contact. As a result of these conflicting outcomes, people continue to question the value of computer technology even as it permeates their daily lives more and more [25].

Given this skepticism, it is important to continue to explore if and how technology can be used to support communication with and awareness of the people we care about. In the last several years, there has been an increased interest in both academia and industry in designing technologies for homes and families (e.g. [9, 15, 19, 20]). Such design offers a number of interesting challenges. A huge diversity of ages, abilities, interests, motivations, and technologies must be accommodated. People are much more concerned about the aesthetics of technology artifacts in their home than at work [27], their values may influence their use of technology [26], and playful entertainment rather than efficiency or practicality may be the goal [6].

As part of the European Union-funded interLiving [13] project, we are working together with diverse families from Sweden, France, and the U.S. to design and understand the potential for new technologies that support communication among diverse, distributed, multi-generational families. Using a variety of established research methods from participatory design, CSCW, and ethnography, as well newer methods involving cultural probes [7] and our own technology probes, we have learned about the needs and desires of the families, introduced them to new types of technology, and supported them in becoming partners in the design of new technologies.

BACKGROUND

One of the key objectives of the interLiving project is to experiment with different design methodologies. Each of the authors' organizations has long-standing experience in participatory design [24], which remains the core strategy for the project. However, we each have different experiences and perspectives. Families, and the individuals within them, represent a new user group for all of us. interLiving provides us with the opportunity to examine our differences, draw from our collective backgrounds, and integrate the most effective approaches.

One of our challenges is to develop new participatory design strategies in which family members can actively participate in the design of new technology. A typical HCI approach would be to interview the families, create a design, develop the technology and then test it to see what the families like or do not like. However, we would like to come up with methods that enable families to more directly inspire and shape the technologies that are developed. Our hypothesis is that this will lead to designs that will work better in the long run because they address families' needs and/or desires better.

We do not expect the family members to become designers, but we do want them to be active partners in the design process. If we only use the typical HCI strategy described above, we believe it might discourage active participation by users, as the design concept is already well established by the time the users see it. Their suggestions are likely to relate to details about the user interface and will not be fundamental contributions to the technological design [4].

Our original proposal for interLiving was to distribute 'seeding' technologies into the families' homes, to provide families with ideas about what we would like to develop. We expected family members to critique these technologies and provide us with feedback that would affect our subsequent designs. As the project progressed, we shifted to the concept of a 'technology probe.'

DEFINITION

A probe is an instrument that is deployed to find out about the unknown - to hopefully return with useful or interesting data. There is an element of risk in deploying probes; they might fail or bring unexpected results. In the interLiving project, we chose to use probes to study families because the complex personal and private environments they live in makes it challenging to learn about their needs and attitudes towards technology using conventional HCI techniques.

Technology probes are a particular type of probe that combine the social science goal of collecting information about the use and the users of the technology in a real-world setting, the engineering goal of field-testing the technology, and the design goal of inspiring users and designers to think of new kinds of technology to support their needs and desires. A well-designed technology probe should balance these different disciplinary influences.

On the social science side, technology probes reject the strategy of introducing technology that only gathers 'unbiased' ethnographic data. We assume that the probes will change the behavior of our users - in our case, the character of inter-family communications. On the other hand, we recognize the benefits of collecting data in-situ - we were interested in observing how families' communication patterns and their interpretation of the technology changed over time. On the engineering side, technology probes must work in a real-world setting. They are not demonstrations, in which minor details can be finessed. Therefore, the main technological problems must be solved for the technology probes to serve their purpose.

On the design side, technology probes are similar to Gaver et al.'s cultural probes [7] - kits of materials such as disposable cameras and diaries meant to inspire people to reflect on their lives in different ways. A number of researchers, including ourselves, have used cultural probes to elicit both design inspiration for new domestic technologies and information about the users of such technologies [8, 26]. However, cultural probes tend to involve a single activity at a particular time and are not necessarily technologies themselves. Dunne and Raby's Placebo Project [5] is closer to the concept of a technology probe: they introduce thought-provoking technologies into people's homes for periods of time. However, they do not use the technology to collect data about its own use.

Our technology probes involve installing a technology into a real use context, watching how it is used over a period of time, and then reflecting on this use to gather information about the users and inspire ideas for new technologies. A well-designed technology probe is technically simple and flexible with respect to possible use. It is not a prototype, but a tool to help determine which kinds of technologies would be interesting to design in the future. A successful technology probe is open-ended and explicitly co-adaptive [18]: we expect the users to adapt to the new technology but also adapt it in creative new ways, for their own purposes.

In addition to the technology itself, a successful technology probe requires analysis and reflection about its use during and after the deployment by both researchers and users. There are many ways this could be accomplished, but we selected three based on our previous research experiences and areas of expertise.

From a social science perspective, we were interested in learning how families communicate with each other and how the probes helped or hindered their ability to do so. We used ethnographic interviews with the families in their homes before and after the deployment to gather this information. From an engineering perspective, we were interested in how and by whom the probes were used to support communication, so we instrumented them to log things like dates, times, and actions so that we could reconstruct the usage over time.

Finally, from a design perspective, we were interested in seeing what ideas the probes would inspire for new technologies. Our background in participatory design suggested that low-tech prototyping workshops [24] could help elicit creative ideas. We provided the families with art supplies like paper, clay, and pipe cleaners and asked them to build new communication technologies, inspired by positive and negative scenarios that some of them encountered using the probes.

DISTINGUISHING FEATURES

Technology probes can be distinguished from prototypes or products as follows:

Functionality: Technology probes should be as simple as possible, usually with a single main function and two or three easily accessible functions. Prototypes may have many layers of functionality and address a range of needs, not all of which may even be implemented.

Flexibility: Although technology probes should not offer many functionality choices, they should be designed to be open-ended with respect to use, and users should be encouraged to reinterpret them and use them in unexpected ways. Prototypes are generally more focused as to purpose and expected manner of use.

Usability: Technology probes are not primarily about usability in the HCI sense. They are not changed during the use period based on user feedback. In fact, a deliberate lack of certain functionality might be chosen in an effort to provoke the users. For prototypes, usability is a primary concern and the design is expected to change during the use period to accommodate input from users.

Logging: Technology probes collect data about users and help them (and us) generate ideas for new technology. Logging allows researchers to create visualizations of the use of the probes, which can be discussed by both users and designers. Prototypes can collect data as well, but this is not a primary goal.

Design phase: Technology probes should be introduced early in the design process as a tool for challenging pre-existing ideas and influencing future design. Prototypes appear later in the design process and are improved iteratively, rather than thrown away.

IMPLEMENTATION

In the interLiving project, we have discussed developing a variety of technology probes. Such probes can be used by individuals, groups of family members or everyone in the family. They may be designed for the home or settings outside the home. They may be fixed or mobile, hard-wired or wireless, large or small, new or existing. The main criteria is that they be different enough from commonly available technologies that they provoke families to consider how they do or don't fit into their lives.

We have developed and installed two technology probes: the messageProbe and the videoProbe, described in the next two sections. Each was designed to gather data about a family's communication patterns while inspiring them to think about new ways of communicating. These probes are not new technologies from a research perspective, but they are novel from the perspective of many families, many of whom may equate technology with desktop computers.

In the deployment of both probes, we ran into a number of technical and logistical roadblocks. We encountered service and administrative problems getting high-speed Internet access installed in some of the families' homes, as well as breakdowns of our own hardware and software, requiring additional visits to the families' homes to correct the problems. Despite these problems, we were able to successfully deploy the probes in families' homes for a month or longer. We offer these problems as cautions to other researchers, but believe they can be avoided or minimized in the future.

Figure 1. messageProbe

MESSAGE PROBE

The messageProbe is a simple application that enables members of a distributed family to communicate using digital Post-It notes in a zoomable space (Figure 1). It can function synchronously, with two or more family members writing and drawing from different locations at the same time, or asynchronously, with family members checking it periodically for new messages from other households. The probes are connected only to a small set of family members, removing the need for complicated setup and remembering names, addresses, or buddy lists. There is no mouse or keyboard – just a writable LCD tablet and pen.

Hardware and Software

The messageProbe software was built using Java and three Java-based toolkits: the University of Maryland's Jazz, Sun's Java Shared Data Toolkit 2.0 (JSDT), and Interbind's XIO, all available for download [1, 12, 14]. The hardware requirements include a writable LCD display, such as Wacom's PL 500 Series, or a regular graphics tablet and a monitor. The software runs on the Windows and Macintosh OS X platforms.

Design

The messageProbe builds on work from three fields. First, the technology is influenced by shared whiteboard projects for use in the workplace [21] and recent attempts to bring such technology into the home, such as the Casablanca project's ScanBoard [9]. Second, in an effort to keep remote family members connected, we were also influenced by research in remote awareness [3]. Finally, our interface design is based on zoomable user interfaces [1]. We also did a lot of work to make the visual presentation and interaction as minimal as possible so the application would feel more like the simple paper notes it was based on and less like a complicated computer with buttons and icons.

We decided to build the messageProbe based on virtual notes because of the popularity of paper sticky notes for informal family communication. We lost the ability to stick notes on anything anywhere in the house, but gained an unlimited supply of notes and the ability to share them remotely with other family members.

With the potential for multiple remote family members to be viewing, manipulating, and writing on their devices simultaneously, there were a number of usability and synchronization issues to consider. Not only do family members at multiple locations share the message space, but also multiple family members at the same location share a single message creation and viewing device.

Thus, we chose to implement a bulletin board-like interface. All users share control of the notes in the message space. Anyone can write on or move a note in the space, regardless of who created it. New notes are immediately sent to all the devices in the family and are displayed in the same location on all devices. We did not want to force an organization of notes on users, but needed some way of arranging them initially. Notes are arranged according to their creation time in a grid, with older notes pushed higher and made smaller.

Organization of notes beyond the default placement is up to users. Notes can be dragged out of the message grid anywhere in the message space. Notes can also be dragged back into the grid, where they resume their place in the time-based order. As notes are added or removed from the grid, the grid reorganizes itself to fill up space. This design allows for some interesting interactions, which add to users' sense of remote awareness. Two users can draw on the same note at the same time or one user can move a note that someone else is writing on.

There is no delete function – users add to existing notes, create new ones, and move old ones. Our first design included these features, plus time and date information for each message. However, we wanted the probe to feel different from a 'regular' computer, so we took away common visual computer signs, like title bars, borders, bad typography, symbols to click on, etc. After much design work and several iterations, there were no longer any complicated interactions or dialog boxes.

Users simply tap a virtual pad of notes to create a new one, and then write on it. Tapping on a note other than the one that currently has focus zooms the focus to the other note. Tapping outside a note zooms the space out to show all the notes. At the first demonstration of the messageProbe in Sweden to the Swedish families, three-year-old Vera simply started to draw on it, just as if it was paper and crayon. No instructions whatsoever were needed.

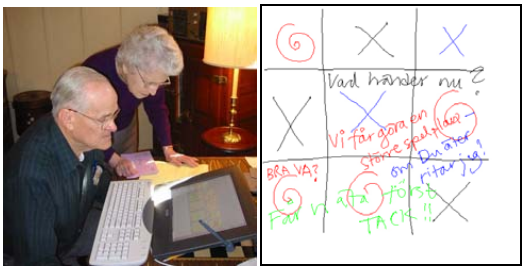


Figure 2. U.S. messageProbe (left) and Swedish message (right). (Note that the keyboard was not used for the messageProbe.)

Probe Deployment – U.S. Family

We deployed the probe in the three households of our U.S. family design partners for 6 weeks in early 2002 (Figure 2, left). These households included a nuclear family with two parents and two school-age children, and two sets of grandparents, all living within 15 km of each other. We provided computers and high-speed Internet access to both sets of grandparents; the nuclear family already had both. While we wanted to provide all of the households with a writable LCD tablet, we only had one of these devices. One set of grandparents used this device, while the other households used graphics tablets and monitors.

For all of the deployments, we wanted to be able to place the probes in high-traffic areas of the families' homes, where family members would hopefully look at them and use them often. We were relatively successful in doing this, but we had to respect the families' wishes and compromise in some cases. In the nuclear and maternal grandparent homes, the messageProbes were located in the kitchen and main living areas, respectively, both high-traffic areas. In the paternal grandparents home, the probe was placed in the basement, which was a bit more out of the way.

The family created over 120 messages during the trial, and in all of the households, someone checked the probe at least once a day. The messages were almost exclusively text, with few drawings or doodles. The two grandfathers wrote the most notes, followed by the father. The two children wrote a few notes each and the grandmothers and the mother wrote one or two each. The two sets of grandparents didn't communicate with each other directly - they each just wrote notes to the nuclear family, despite the fact that everyone could see all the notes. When we interviewed the families about this, we found that this lack

of direct communication was typical. The grandparents got along, but did not need to communicate with one another often.

Status updates were the most numerous types of notes, but many of these had to do with technology problems. Notes about minor news, feelings, and coordination were nearly as numerous, while there were also a few questions and reminders. The only one who used the probe in the nuclear household regularly was the father. The children were frequently too busy, and the mother preferred the phone. The paternal grandparents had no prior computer experience, but the simple interface of the messageProbe provided a good introduction. The lack of a delete function made the grandfather self-conscious about mistakes, so he wrote many of his notes on paper first. The maternal grandparents had the most trouble with the probe. They required a new modem, a new IP address, and had a problem with their pen due to electrical interference.

Many of the family members wanted a notification function, such as an audio cue, for new messages. All the grandparents were disappointed that the grandkids didn't use it more, but the probe helped reveal that coordination between the nuclear household and the grandparents for childcare was an important issue. However, everyone felt that it was not reliable enough for important communications. It was fun, but the phone was better for a quick response.

Probe Deployment – Swedish Family

In Sweden, the messageProbe was installed in two households of one family over several months during the summer of 2002. We provided both households with LCD tablets and Apple Cubes. The households included two sisters, one living with her boyfriend and the other with her husband and two small children.

The first sister and her boyfriend lived in a small apartment and placed the probe in their bedroom, next to their computer. This was a high-traffic area, but they chose to switch the probe off at night because of the noise and light it generated. The second sister and her family placed the probe on an unused dining table in the downstairs of their house. The probe was visible from nearly every room downstairs because of the open floor plan in the house.

This family wrote over 200 notes during the course of the trial. There was considerable difference between how much the sisters used it vs. their husband and boyfriend. The sisters treated it as a natural continuity of how they already communicate - a constant flow of notes, with text and drawings, answering machine messages and telephone calls. Their use of the messageProbe was another way of leaving notes. By contrast, their husband and boyfriend did not communicate with either each other or their spouses to the same extent, and did not use the probe as often.

In contrast to the U.S. family, the Swedish messages were more playful (Figure 2, right). One sister played remote 'connect-the-dots' with her niece. The two children enjoyed the probe so much that at times they fought over the pen. For the adults, messages were often annotated repeatedly from both sides. When there was no more space to write, they continued on another note.

Like the U.S. family, the Swedish family discussed a visual or audio cue to provide awareness when someone on the 'other side' was writing a message. However, they also noted that there was a negative side to such a signal because it could be distracting or annoying if you were occupied with other things. They had similar technical problems with the probe not working at times during the trial, and the zooming feature on their computers was rather slow. In spite of the problems though, they all enjoyed it and said that it actually added a new dimension to their relationships.

Conclusions

Despite problems with robustness, the probes were helpful in revealing communication patterns and technology needs and desires. Many of the messages in the U.S. family involved attempts at coordination for things like picking up children, indicating that this is a promising area of research for new technologies. In addition, the playful use of the probe by the Swedish family indicated a desire for simple, fun ways of providing remote awareness between households. The probes also revealed more subtle aspects of communication in the families that would likely not have come up in interviews – i.e. the unique communication habits of the Swedish sisters and the U.S. grandparents.

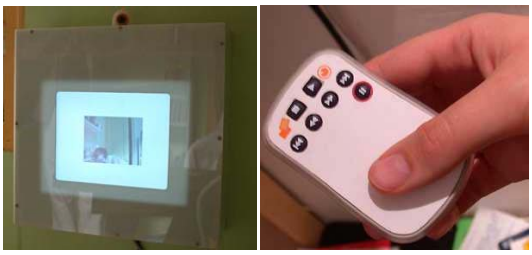


Figure 3. videoProbe (left) and customized remote control (right)

VIDEOPROBE

The videoProbe (Figure 3, left) provides a simple method of sharing impromptu images among family members living in different households. We use a video camera that takes a snapshot when the image it captures becomes steady for approximately three seconds. The images are stored and made available to anyone else in the network. Family members can browse the images with a remote control (Figure 3, right). Images fade over time and eventually disappear, to encourage families to create new ones.

Hardware and Software

The videoProbe consists of an Apple Cube, a Wacom PL-500 LCD tablet, a Philips ToUCam Pro USB camera, a pair of Apple USB speakers, a Keyspan Digital Media remote

control, a USB hub and an Apple Airport base for wireless networking. We selected the Apple Cube both for its unconventional look and its silence (it has no cooling fan). Even so, some families complained about the hard drive being noisy. The screen/tablet is used only for display. The Airport base allowed us to install the videoProbe just about anywhere in the families' homes. The software is implemented in C++ with the videoSpace toolkit [23].

Design

The videoProbe was inspired by research on media spaces [2], which demonstrated the power of video to support remote awareness. We have chosen to share still images rather than live video for several reasons that relate to the goals of technology probes. First, real-time video would have been difficult to achieve in a home installation. Second, still images support asynchronous as well as almost synchronous communication [3]. Third, the design requires family members to interact with the probe, giving us a way to capture usage data and discuss their patterns of use.

Considering the variety of devices and cables involved in the videoProbe hardware, we had to develop a packaging design that was compact, non-intrusive and simple to handle. We structured the technology into two units: the computer and its power supply and a customised rectangular box that houses the screen and the rest of the equipment. These units are connected via a covered lead, which includes the video, power and USB cables.

The videoProbe was designed to be usable in a variety of spatial configurations within the families' homes. The box can stand alone on any item of furniture. A hole in the back allows it to be mounted onto a wall, like a picture frame. The unit may also lie flat on its back, so that it can be used for message/drawing applications.

We designed the display unit to exploit the high quality of the screen and video camera. At full resolution, the images do not fill the screen, so we covered the remaining parts of the screen and the rest of the box with white plastic. We wanted to keep the visual design as simple as possible, to blend in with any decor. The white plastic does not attract much attention and naturally disappears into its surroundings when the system is not active. When a family member approaches the videoProbe, the video fades in and highlights the packaging with a glowing white semi-transparent band, emphasizing the reactiveness of the unit.

The camera sits on top of the videoProbe screen, similar to a webcam on a monitor. We wanted family members to be able to point the camera in any direction, so we created a notch filled with foam on the top of the videoProbe. This makes it easy to lift up the camera, rotate it, and fix it into the desired position. The camera can be focused by hand and has a wide range, including objects that are only millimeters away. We provided a long cable, housed inside the box, to enable family members to take the camera out of the videoProbe to take close up shots of things nearby.

To simplify the use of the videoProbe, we created a custom-made graphic design for the remote control. Our earlier tests showed that even the few tasks executed by the remote control could be confusing. It was not obvious how to put an image into or remove it from the album, and these actions are not clearly related to culturally-established VCR control iconography, such as <<, >, >>. Note that users also face these problems when attempting to manipulate stored images on commercial digital cameras.

Figure 4. videoProbes in the French families' homes

Probe Deployment – French Families

We installed four videoProbes in the homes of two French families during the summer of 2002. The first pair of videoProbes was installed in the homes of two sisters, both living in Paris (Figure 4, left). The first sister designed a kind of 'media wall' for the probe in the corridor of her flat, due to the lack of space in the apartment.

The corridor was designed as a substitute for a social lounge area and the videoProbe fit very well into this environment. The second sister and her roommates let us drill a hole so we could place the videoProbe on the wall. They also moved things around and were interested in finding a location that was integrated into their living space. Unfortunately, she had to move soon after we connected the probe so we could only collect limited data.

The second pair of videoProbes was installed in the homes of two brothers, both living in suburbs of Paris about 20 km apart. These families decided that they wanted to place the videoProbes in the main living area, where they could be seen from both the sofa and the dining room table. Unlike the two sisters, their settings were more formal and we could not hang the probes on the wall. Instead, the families placed them on tables or sideboards, rearranged to accommodate plants, vases, and lamps (Figure 4, right).

Preliminary observations of the use of the videoProbes already show a variety of patterns of use. Kids and young adults like to use it in a playful way, e.g. sending pictures where they make faces or taking strange close-ups. They also use it for communication, e.g. taking a picture of a hand-written message. We expect these patterns to evolve when the probes are used over a longer period of time and become more integrated into the families' lives.

EMERGING DESIGNS

Our experiences deploying the messageProbe and the videoProbe in the homes of our family design partners has led us to two promising areas of research. Through log

files, interviews, and workshops, the families have identified a variety of different interests, from practical to whimsical, for staying in touch with members between and within households. We are developing two types of prototypes that reflect this diversity: some to support family coordination and some to support playful interaction.

In addition, we have realized that families need a far better method of specifying with whom they communicate. To meet this need, we are exploring different approaches that will be integrated into our prototypes. Finally, our experience installing the probes to fit around existing objects in the home suggested that we should explore applications that take advantage of existing objects. We are designing some of our prototypes to address this need, by studying which objects in the home can be augmented to support coordination and playful interaction.

Family Coordination

One conclusion that we and our design partners drew from the technology probe installations was that coordination between and within households is important but difficult. Different family members have different coordination needs, and everyone makes use of different methods and tools. One workshop we held with the U.S. family following the messageProbe deployment was particularly useful in allowing them to reflect on this problem.

The goal of the workshop was to generate ideas for family communication and coordination technology, based on experiences with the probe. We motivated the discussion by discussing examples and events of coordination scenarios and breakdowns that we had learned about through the messageProbe trial. We split the family into teams and gave them low-tech prototyping art materials (colored paper, string, clay, etc.) to use to design technology solutions for the scenarios.

The mother and father wanted to keep track of everyone's schedules. They built shared calendars embedded in the refrigerator and added features to their cell phones to connect them with this calendar. Their use of the messageProbe was focused on coordinating their childrens' activities and getting help with this from the grandparents, and their prototypes reflected this need as well.

The grandparents wanted to keep track of people. They built key hooks by the door that noted who was home, and a ring that pinched the wearer if someone wanted to talk to them. Their use of the messageProbe was marred by technology breakdowns and by a preference for pen and paper over graphics tablets, and their devices reflected their desire for something simpler and more direct.

The kids designed small devices for keeping in touch with friends and parents – voice activated key chains for sending messages and watches that displayed after-school schedules. They didn't use the messageProbe much at all, saying that they were frequently too busy or not home. They wanted devices they could carry with them and use wherever they were.

Overall then, staying connected with and aware of family was important, but people had different motivations for doing so and wanted to do it in different ways. As a first step to supporting them, we are developing new coordination interfaces to enable households to view each other's schedules and to leave messages for one another. Later, we could extend this service to improve communication, portability, and tracking by supporting GPS-equipped PDAs, cell phones, and other small devices.

Family Playfulness

Another conclusion that became clear after the deployment of both the probes is that families want to have fun together, even at a distance. With the messageProbe, we saw tic-tac-toe boards, connect-the-dots games, and family member caricatures, all bringing family members from different households into shared, playful activities. With the videoProbe, early interactions included family members making funny faces at each other at a distance.

This is not a startling conclusion – Huizinga coined the term Homo Ludens in 1950, defining humans as playful creatures [10]. However, aside from games, the design of technologies has generally focused on tools to improve our efficiency, not to support our playful side. It is only recently that designers such as Gaver have begun to think about how to design to support playfulness [6]. Our technology probes were built to be open-ended and ambiguous to inspire new uses. The fun way our design partners interacted with them seems to validate the playful side. We are currently working on prototypes that build on these ideas.

CONCLUSIONS

We believe that technology probes are a promising new design tool for working with families as partners in the design of new technologies. Despite the technical difficulties encountered during the deployment of the messageProbe and videoProbe, we believe that as technology probes, they were successful in three ways.

First, they helped reveal practical needs and playful desires within and between distributed families. Second, they provided real-life use scenarios to motivate discussion in interviews and workshops. Finally, they introduced families to new types of technologies beyond the accustomed PC-monitor-mouse-keyboard setup, which we believe encouraged them to consider more whimsical and creative uses of technology in our design workshops.

ACKNOWLEDGMENTS

We would like to thank our family design partners for all their work on this project. The interLiving project is supported by EU IST FET, through the Disappearing Computer Initiative.

REFERENCES

1. Bederson, B., Meyer, J. & Good, L. (2000). Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *Proc. of UIST 2000*. ACM Press, pp. 171-180.
2. Bly, S., Harrison, S. & Irwin, S. (1993). Mediaspaces: Bringing People Together in a Video, Audio and Computing Environment. *CACM, 36(1)*, pp. 28-47.
3. Dourish, P. & Bly, S. (1992). Portholes: Supporting Awareness in a Distributed Work Group. *Proc. of CHI '92.* ACM Press, pp. 541-547.
4. Druin, A. (2002). The Role of Children in the Design of New Technology. *Behaviour and Information Technology*, 21(1), pp. 1-25.
5. Dunne, A. & Raby, F. (2001) *Design Noir: The Secret Life of Electronic Objects*. Switzerland: Birkhauser.
6. Gaver, W. (2002). Designing for Homo Ludens. *i3 Magazine*, June (2002), pp. 2-5.
7. Gaver, W., Dunne, A. & Pacenti, E. (1999) Projected Realities: Conceptual Design for Cultural Effect. *Proc. of CHI '99.* ACM Press, pp. 600-608.
8. Hemmings, T., Crabtree, A., Rodden, T., Clarke, K. & Rouncefield, K. (2002). Probing the Probes. *Proc. of PDC 2002*. CPSR, pp. 42-50.
9. Hindus, D., Mainwaring, S., Leduc, N., Hagstrom, A. & Bayley, O. (2001). Casablanca: Designing Social Communication Devices for the Home. *Proc. of CHI '01.* ACM Press, pp. 325-332.
10. Huizinga, J. (1950). *Homo Ludens: A Study of the Play Element in Culture*. Boston: The Beacon Press.
11. Hutchinson, H., Plaisant, C. & Druin, A. (2002). Case Study: A Message Board as a Technology Probe for Family Communication and Coordination. Position Paper, Workshop on New Technologies for Families, *CHI '02,* http://www.cs.umd.edu/hcil/interliving/chi02.
12. Interbind (2002), http://www.interbind.com.
13. Interliving (2002), http://interliving.kth.se.
14. Java Shared Data Toolkit (2002), http:// java.sun.com/products/java-media/jsdt/index.html.
15. Kidd, C., Orr, R., Abowd, G., Atkeson, C., Essa, I., MacIntyre, B., Mynatt, E., Starner, T. & Newstetter. (1999). The Aware Home: A Living Laboratory for Ubiquitous Computing Experience. *Proc. CoBuild 99*.
16. Kraut, R., Kiesler, S., Boneva, B., Cummings, J., Helgeson, V. & Crawford, A. (2002). Internet Paradox Revisited. *Journal of Social Issues, 58(1)*, pp. 49-74.
17. Kraut, R., Mukhopadhyay, T., Szczypula, J., Kiesler, S. & Scherlis, W. (1998). Communication and Information: Alternative Uses of the Internet In Households. *Proc. of CHI '98*. ACM Press, pp. 368-374.
18. Mackay, W. (1990). Users and Customizable Software: A Co-Adaptive Phenomenon. Ph.D. Thesis. Massachusetts Institute of Technology.
19. McClard, A. & Somers, P. (2000). Unleashed: Web Tablet Integration into the Home. *Proc. of CHI 2000*. ACM Press, pp. 1-8.
20. Mynatt, E., Rowan, J., Jacobs, A. & Craighill, S. (2001). Digital Family Portraits: Supporting Peace of Mind for Extended Family Members. *Proc. of CHI '01.* ACM Press, pp. 333-340.
21. Pederson, E., McCall, K., Moran, T. & Halasz, F. (1993). Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *Proc. of CHI '93*. ACM Press, pp. 391-398.
22. Putnam, R. (2000). *Bowling Alone*. New York: Simon & Schuster.
23. Roussel, N. (2001). Exploring New Uses of Video with videoSpace. *Proc. of EHCI '01*. Springer, pp. 73-90.
24. Schuler, D. & Namioka, A. (1993). *Participatory Design: Principles and Practice*. New Jersey: Lawrence Erlbaum.
25. Shellenbarger, S. (2001). Work & Family: Americans Are Split on Impact of Technology on the Family. *The Wall Street Journal*, January 10, 2001.
26. Voida, A. & Mynatt, E. (2002). Grounding Design in Values. Position Paper, Workshop on New Technologies for Families, *CHI '02,* http://www.cs.umd.edu/hcil/interliving/chi02.
27. Westerlund, B. & Lindquist, S. (2002). Aesthetic Perspectives on Participatory Design in the InterLiving Project. Position Paper, Workshop on New Technologies for Families, *CHI '02,* http://www.cs.umd.edu/hcil/interliving/chi02.
28. Westerlund, B., Lindquist, S. & Sundblad, Y. (2001). Cooperative Design of Communication Support for and with Families in Stockholm - Communication Maps, Communication Probes and Low-Tech Prototypes. *Proc. of Equator IRC Workshop on Ubiquitous Computing in Domestic Environments*.

Artful Media

Doree Duncan Seligmann
Avaya Labs

Proximity as an Interface for Video Communication

Nicolas Roussel, Helen Evans, and Heiko Hansen
LRI & INRIA Futurs

When you're far away, your own image reflects in it like in a distorting mirror, blurred and imprecise. As you move toward it, however, it becomes clearer and more accurate. By the time you reach it, the reflection is almost perfect, as in a conventional mirror. What you see is not a simple optical reflection but a video image captured, processed, and displayed in real time.

What appears to be a mirror is in fact an interactive video communication system—MirrorSpace—connected to similar devices in other places. As someone passes in front of one of the other devices, a vague silhouette appears on the one in front of you, mixed with your own image (see Figure 1). As he approaches, his image becomes sharper, allowing you to recognize him. When he reaches the device, his face gets fully merged with yours, letting you look into each other's eyes.

Evolution of the project

MirrorSpace was originally conceived as a prototype for the interLiving project (http://interliving.kth.se/) of the European Disappearing Computer initiative (2001–2003). This project focused on the design of technologies to support communication among family members located in different households. For three years, three Swedish and three French families collaborated with a multidisciplinary research team with expertise in computer science, social science, and design.

One of the technologies we investigated in this project was the exchange of still images or video streams between households. Previous work on video, including our own, has shown that it's well suited for coordination and informal communication.[1] However, traditional video technologies designed for work settings don't necessarily fit in home settings where space can be tight and serve multiple purposes, and where the relationships between family members can be complex. Our work on MirrorSpace started as we got interested in the use of space and distance in video-mediated communication.

(Un)use of space in video-mediated communication

One of the advantages of video over audio or text-based systems is the ability to transmit nonverbal information. However, while many studies have focused on eye gaze and gesture in video-mediated communication, little work has been carried out on proxemics,[2] one of the most fundamental elements of nonverbal communication. See the "Proxemics" sidebar for an overview.

Physical proximity to other people is a form of communication that we all use, although we're barely aware of it. Space and distance let us define and negotiate the interface between private and public, particularly during the moments leading up to contact. By altering our physical distance from other people in a space, we communicate subtle messages—such as our willingness to engage into dialogue with them, the desire for more intimacy, or a lack of interest. For each communication situation, we have a distance that we find appropriate. Certain feelings

Figure 1. Images from an early MirrorSpace concept video (August 2002).

1070-986X/04/$20.00 © 2004 IEEE

Published by the IEEE Computer Society

Proxemics

The term *proxemics* refers to the study of spatial distances between individuals in different cultures and situations. It was coined by E.T. Hall in 1963 when he investigated man's appreciation and use of personal space. Hall's model[1] lists four distances that North Americans use in the structuring of personal dynamic space: intimate (less than 18 inches), personal (between 18 inches and 4 feet), social (between 4 and 12 feet), and public (more than 12 feet). For each communication situation, we have a distance within these four categories that we find appropriate. If the perceived distance is inappropriate, we become uncomfortable and usually adjust it by physically moving closer or farther away, or even simply turning our head or looking in another direction.

Reference

1. E.T. Hall, *The Hidden Dimension: Man's Use of Space in Public and Private*, Doubleday, 1966.

Daniel Rozin's Mirrors

Daniel Rozin has created several installations that use image processing techniques to turn a set of motorized, nonreflective surfaces into a virtual mirror. His *Wooden Mirror*, for example, is made of 830 square pieces of wood lit from above that can be tilted up and down individually, appearing lighter or darker depending on the angle. The whole array can thus display a rough reflection of whatever is in front of it (Figure A). *Trash Mirror* is a similar Rozin installation, made of 500 irregular pieces of trash collected on the streets of New York. Yet another piece, *Shiny Balls Mirror*, consists of 900 hollow metal tubes with polished chrome balls placed in them. Here, the brightness of each "pixel" is controlled by moving the ball in (darker) or out (brighter) of the tube. The display thus serves as a mirror in two ways: as a whole, but also as it reflects the viewer 900 times on the shiny balls.

Figure A. Daniel Rozin's Wooden Mirror.

or emotions, for example, are difficult to share unless the two partners are close.

Existing systems for video-mediated communication fail to take proxemics into account. Although some of the people who design the systems understand the importance of proxemics, they fail to give it much consideration or to provide the support it requires. Systems are usually designed for a specific task, corresponding to a certain interpersonal distance. Physical constraints often make it impossible for people to come closer to the device than expected or to move away from it.

MirrorSpace: Design concept

Our work on MirrorSpace focuses on creating a video communication system that takes physical proximity into account. We're particularly interested in how people's interactions can trigger smooth transitions between situations as extreme as general awareness of remote activity (where anonymity is preserved) to close and intimate forms of communication. As the name suggests, MirrorSpace relies on a mirror metaphor. This system's key characteristics include the original placement of the camera combined with translucently overlaying the images and using a proximity sensor combined with a blur filter.

As a cultural artifact, the mirror has a prominent position in the creation and expression of aesthetics. Throughout Western culture, in narratives such as the Narcissus myth, *Snow White*, or *Through the Looking Glass*, the mirror has come to symbolize many things—including vanity, deception, identity, or a passage to another world. Unsurprisingly, numerous artists and designers have picked up on these meanings and taken advantage of the universal and irresistible fascination for self-image to explore the boundaries between the analog and digital worlds. Examples of these works include Christian Möller's *Electronic Mirror* (http://www.christian-moeller.com), Scott Snibbe's *Screen Series* (http://www.snibbe.com), Camille Utterback's *Liquid Time* (http://www.camilleutterback.com), and Daniel Rozin's various mirrors (http://fargo.itp.tsoa.nyu.edu/~danny; also see our sidebar, "Daniel Rozin's Mirrors").

Sometimes we can also perceive a mirror as a surface for mediating communication with its own rules and protocols. As many subway commuters know, making eye contact with a stranger through a reflecting surface is usually considered less intrusive than direct eye contact. Because humans already associate the mirror with this idea of reaching out to other people and other spaces,

Figure 2. Looking into each other's eyes.

looking into someone's eyes, the viewer has to look at the camera and thus can no longer see where the other person is looking.

MirrorSpace superimposes the live video streams from all the connected places on a single display on each site so that people see their own reflection combined with the ones of the remote persons. We felt it was important for people to actually look into each other's eyes and possibly merge their portraits into one, so we decided to place the camera on the screen, rather than beside it. This setup allows participants to come close to the camera while still being able to see the remote people and interact with them (see Figure 2).

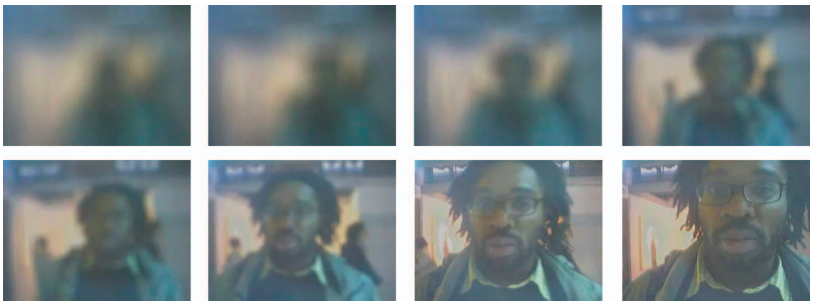


Figure 3. From peripheral awareness to close communication by moving toward the device.

Boyle et al.[4] showed that a blur filter is an effective way of masking out potentially sensitive information in an always-on video link. They also proposed to adapt the blur level to the distance between the user and the communication device, although their system only used three different levels. In contrast, instead of creating a series of shared spaces corresponding to particular interpersonal distances, MirrorSpace aims to create a continuum of space that will allow a variety of interpersonal relationships to be expressed.

MirrorSpace includes a proximity sensor that measures the distance to the closest object or person in front of it. A blur filter is applied on the images to visually express a distance computed from the local and remote sensor values. Blurring distant objects and people lets the up-close viewer perceive distant movement or passing with minimum involvement. It also offers a simple way of initiating or avoiding a change to a more engaged form of communication by simply moving closer (see Figure 3) or farther away.

we believe it's the ideal enabling metaphor for establishing a new communication experience.

Several video communication systems have recently used a mirror metaphor to provide seductive and pleasant-to-use interfaces. As demonstrated by Morikawa and Maesako,[3] this metaphor helps reduce the psychological distance between local and remote participants by displaying them side by side. However, because the camera is usually placed atop or beside the display, the remote people never seem fully engaged and appear to be looking slightly off, in another direction. To give the impression of

Figure 4. MirrorSpace installation at Mains d'Oeuvres (Paris, May 2003).

MirrorSpace installations

Several pairs of MirrorSpace prototypes have been built and presented to the public as an interactive video installation in four art exhibitions in February, May (Figure 4), July, and November 2003 (Figure 5). These exhibitions gave us the opportunity to observe a large number of people interacting with MirrorSpace in a controlled technical environment. Each of these exhibitions was also an occasion to

refine the prototypes' design and explore new software or hardware possibilities.

Each prototype is made of a thin-film technology liquid-crystal display flat screen, a universal serial bus camera, an ultrasonic proximity sensor, and a computer that runs dedicated software. We designed the prototypes to minimize their technological appearance. The computer and all the wires are hidden from users. The screen and its attached sensors are placed into a wooden box, protected by transparent glass (Figure 6). The screen is oriented in portrait mode and part of the protective glass is covered with mirror film to further push the augmented mirror metaphor.

The image sensor and the camera lens are placed on the protective glass that covers the screen, and then connected to the camera's logic board using hair-thin isolated wires running over the glass. The proximity sensor is placed at the bottom of the screen and connected to the computer via a serial interface.

The software uses the videoSpace library[5] to capture images in real time and send them to the other prototypes, along with proximity sensor values. We're able to connect more than two prototypes, although we never did for the exhibitions. The software applies a two-pass incremental blur filter on each image. The resulting images are then flipped horizontally to produce the expected mirror effect and superimposed using OpenGL.

The system uses the distances measured by all the connected prototypes to compute the blur level to apply to each image. We've investigated three computation modes so far: the first mode uses the distance between people and their screen, the second one uses the sum of these distances, and the third one computes a virtual relative distance from them. Although the software lets you specify a different mode for each prototype, the configurations used for the exhibitions always imposed a strict "what you see is what I see" (WYSIWIS) condition.

Initial user reactions

Several hours of video were shot during the exhibitions, showing visitors interacting with the prototypes and what was displayed on the screens. Although the context isn't exactly representative of a remote video communication, a number of observations are worth reporting, as they're probably related to the nature of MirrorSpace itself rather than this particular context.

Although we tried our best to avoid it, a small delay (up to 500 milliseconds) was sometimes perceived between the capture and the display of images. Surprisingly, most people didn't pay attention to it and some liked it: they were running back and forth to play with their own image and see the blur effect in action. Some people even thought the delay had been introduced on purpose. This illustrates the important difference between the technical preoccupations (focusing mostly on function) usually associated with digital video and how users perceive a system like MirrorSpace that focuses on the use of the images and user experience. We discuss this further in the sidebar, "About Video and Time Delay."

Almost all visitors of the exhibitions agreed on one point: interacting with MirrorSpace is fun. Proximity sensing helps create an intimate relationship between users and the system. As we said, many of them played with their own image and the blur effect. People didn't hesitate to make a fool of themselves and many took pic-

Figure 5. MirrorSpace installation at the Centre Pompidou (Paris, November 2003).

Figure 6. Close up showing the proximity (bottom) and image sensors (center).

About Video and Time Delay

Artists like Dan Graham already use time-delay mechanisms in mirror-based installations to let viewers see themselves as both subject and object. (A description of opposing mirrors and video monitors on time delay is available from http://www.sfmoma.org.) We believe that one of the reasons why people weren't bothered by the delay when interacting through MirrorSpace is that it affected both the remote person's image as well as their own simultaneously and was thus immediately perceived and understood. It isn't clear, however, whether the understanding would be the same in the case of a real remote communication.

tures or recorded video clips of themselves and others interacting with the system.

When they saw another person appearing next to them on the screen, many people turned around, looking for that person behind them. This clearly shows that MirrorSpace creates a sense of shared space and that it's perceived as a mirror more than a video communication system. In fact, the majority of the people didn't think about the camera at all. Only after playing with the system for some time did they suddenly ask with surprise, "Where is the camera?"

People who were visiting the exhibitions with friends or relatives tried to overlay their faces. Some went as far as kissing each other. At the same time, other persons were surprised and even disturbed to find strangers able to come so close to them. In that case, they backed away, which made their own image disappear smoothly with the blur effect. This shows that MirrorSpace supports at least part of our accustomed body language.

Directions for future work

One important step for future studies will be the building of other MirrorSpaces. We plan to deploy and demonstrate the system in various other contexts (for example, family households, different buildings of the same research group, and so on). This should help us collect more qualitative and quantitative data about the system's use. In particular, it should be easy to measure the actual time people spend at each distance according to Hall's classification.[2]

We're investigating several technologies that would let us embed the image sensor in the protective glass itself. We're also working on the design of an auditory equivalent to MirrorSpace that could be combined with it in future installations. The challenge here is to design an equivalent to the blur effect that would provide general audible awareness of people far away from the sensor and spoken communication with them as they move closer.

More information on MirrorSpace—including the source code, some images, and videos—is available at http://insitu.lri.fr/~roussel/. **MM**

References

1. W. Mackay, "Media Spaces: Environments for Informal Multimedia Interaction," *Computer-Supported Cooperative Work Trends in Software Series*, John Wiley & Sons, 1999.
2. E.T. Hall, *The Hidden Dimension: Man's Use of Space in Public and Private*, Doubleday, 1966.
3. O. Morikawa and T. Maesako, "HyperMirror: Toward Pleasant-to-Use Video Mediated Communication System," *Proc. ACM Conf. Computer-Supported Cooperative Work* (CSCW 98), ACM Press, 1998, pp. 149-158.
4. M. Boyle, C. Edwards, and S. Greenberg, "The Effects of Filtered Video on Awareness and Privacy," *Proc. Conf. Computer-Supported Cooperative Work* (CSCW 00), ACM Press, 2000, pp. 1-10.
5. N. Roussel, "Exploring New uses of Video with VideoSpace," *Proc. Int'l Federation for Information Processing's Working Conf. on Eng. For Human–Computer Interaction* (EHCI 01), LNCS 2254, Springer, 2001, pp. 73-90.

Contact author Nicolas Roussel at roussel@lri.fr; contact authors Helen Evans and Heiko Hansen at {helen, heiko}@hehe.org.

Contact Artful Media editor Doree Duncan Seligmann at Avaya Labs, 666 Fifth Ave., 11th floor, New York, NY 10103; doree@avaya.com.

Pêle-Mêle, a Video Communication System Supporting a Variable Degree of Engagement

Sofiane Gueddana
LRI (Univ. Paris-Sud - CNRS) & INRIA Futurs *
Bâtiment 490, Université Paris-Sud
91405 Orsay Cedex, France
gueddana@lri.fr

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

ABSTRACT

Pêle-Mêle is a multi-party video communication system that supports a variable degree of engagement. It combines computer vision techniques with spatial and temporal filtering of the video streams and an original layout to support synchronous as well as asynchronous forms of communication ranging from casual awareness to focused face-to-face interactions. This note presents the system's design concept and some of its implementation details.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Computer conferencing, teleconferencing, and videoconferencing

General Terms

Design, Human Factors

Keywords

Video-mediated communication, variable degree of engagement, smooth transitions

1. INTRODUCTION

Video communication systems are most often used for short, synchronous and highly-engaged face-to-face interactions. Previous work on mediaspaces has demonstrated the potential value of long-term video links for casual awareness and informal interaction [4]. Yet, few video systems manage to effectively support both general awareness and face-to-face interactions. Two notable exceptions are Community Bar [5] and MirrorSpace [7], which both provide users with simple ways of choosing the level of engagement that best suits their needs from a discrete (Community Bar) or continuous (MirrorSpace) set of possibilities.

Instant messaging applications make it easy for users to indicate their status and adapt the pace of the conversation to their current context, supporting transparent transitions between synchronous and asynchronous communication. Existing video systems lack this ability to seamlessly transition back and forth between loosely-coupled interactions and highly-coupled ones. We believe that the notions of *variable degree of engagement* and *smooth transitions between degrees* are particularly important for mediated communication and should be taken into account by communication systems designers.

As part of a research project funded by a major telephone company, we are designing a series of image-based communication systems to explore these two notions in the context of the home environment. This work builds on experiences and results from a previous multi-disciplinary project that investigated the communication patterns and needs of distributed families [3, 7]. This project particularly pointed out the importance and difficulty of coordination between and within households, and the need for more subtle, less intrusive forms of communication than the telephone.

This note describes Pêle-Mêle, the first new system we developed. The next section provides an overview of its design concept. We then present some implementation details and conclude with directions for future work.

2. OVERVIEW AND CONCEPT

Pêle-Mêle is a video system designed for between-home close-knit group interaction (e.g. family, friends). It physically consists in a screen equipped with a video camera and connected to a small, unnoticeable computer. The display layout follows a focus-plus-context approach: the screen shows both an overview of all the connected places and a detailed view of the ones where someone is actually communicating through the device. The layout is shared among Pêle-Mêle instances on a strict WYSIWIS basis to help users relate one to another and support gaze awareness.

Pêle-Mêle constantly monitors the activity of local users and classifies it according to a three-level scale: *away*, *available* and *engaged*. The activity observed at each place determines the nature of its on-screen representation, which potentially combines live images and pre-recorded ones that are filtered, delayed or displayed as-is:

away The place is represented by video clips showing past activity and a filtered view of the last image it transmitted.

* projet in|situ| (`http://insitu.lri.fr`), Pôle Commun de Recherche en Informatique du plateau de Saclay (CNRS, Ecole Polytechnique, INRIA, Université Paris-Sud)

CSCW'06, November 4–8, 2006, Banff, Alberta, Canada.

available The place is represented by video clips showing past activity and a delayed live stream.

engaged The place is represented by video clips showing past activity and a live stream which is recorded for later use.

Live images from people engaged in using the Pêle-Mêle are overlaid in the middle of the screen, while available people are shown on the periphery (Figure 1, left). Auditory feedback and smooth animated transitions between these two representations ease perception and understanding of the state changes. Images showing past activity are also displayed on the periphery along a perspective timeline: they slowly shrink and drift toward the center of the screen over time (Figure 1, right).

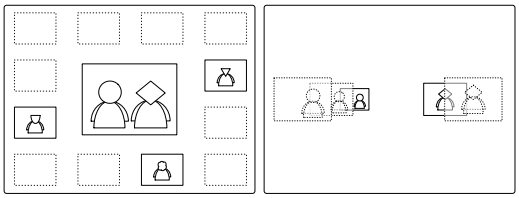
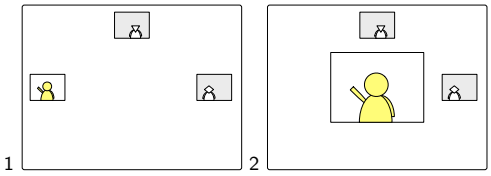


Figure 1: Focus-plus-context view of live streams and perspective timeline effect used for recorded images.

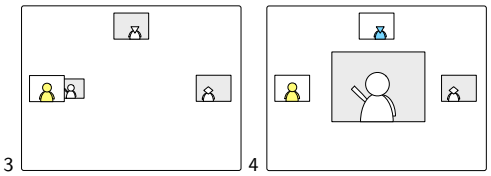
The following scenario further illustrates the concept:

> Joey, Chandler and Ross each have a Pêle-Mêle at home. Joey has some tickets for tonight's game he would like to share with his friends, but Chandler and Ross are not there. Joey waves at the Pêle-Mêle, which switches from *available* (1) to *engaged* (2). His video stream is automatically recorded while he shows the tickets to the camera.

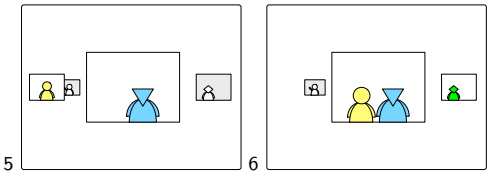


> Joey goes back to his comfortable armchair and favorite TV show, which triggers a transition back to *available* (3). The clip that shows him with the tickets has been added to the display. It slowly drifts in perspective over time and is automatically played in the focus area from time to time. Chandler comes home. His Pêle-Mêle switches from *away* to *available*. Chandler notices Joey's clip as it is played in the focus area (4).

> Chandler now wants to talk to Joey about the tickets. He moves towards the Pêle-Mêle, which switches to *engaged* (5). Joey gets up and moves closer



> to his Pêle-Mêle, which also switches to *engaged*. Their video streams are now superimposed (6) and an audio connection is automatically set up. At the same time, Ross comes home, which switches his Pêle-Mêle to *available*.



3. IMPLEMENTATION DETAILS

Pêle-Mêle is implemented in C++ on an Apple Mac mini computer. It uses the Núcleo[1] toolkit for video capture, recording and transmission as well as simple presence and motion estimation. OpenCV[2] is used for more complex computer vision techniques such as face detection or optical flow computation. Finally, Pêle-Mêle implements spatial and temporal filtering techniques similar to those proposed by Hudson & Smith [2] or Gutwin [1]. These filters are used, for example, to degrade or delay images to mitigate privacy concerns, or to compose them over time to increase the understanding of each other's activities.

Presence is detected by subtracting a reference image from the current one. Motion is estimated by comparing successive images. More robust techniques based on optical flow computation have also been implemented. However, simple image difference is considerably faster and accurate enough for our purpose. We use OpenCV's face detector to estimate the distance that separates people from the device. This assumes a "standard" face size, which produces incorrect estimations for people who don't fit that standard (e.g. children), and works best for people facing the camera at a close distance. Nevertheless, under these particular conditions, it is pretty reliable.

Presence, motion and distance estimations are used to constantly assess the local activity level (Figure 2). Transitions between levels trigger auditory feedback and slow animated transitions on the display that add some hysteresis into the system. Though the chosen computer vision techniques are not particularly stable or robust, they seem to be adequate. Informal testing indicates that users quickly understand how the system works and how they can adjust their level of engagement through simple movements.

We will now describe more precisely the operation modes corresponding to each activity level.

[1] `http://insitu.lri.fr/~roussel/projects/nucleo/`
[2] `http://www.intel.com/technology/computing/opencv/`

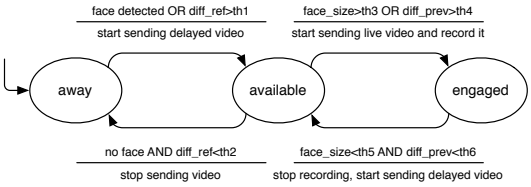


Figure 2: Simplified view of the activity detection algorithm.

3.1 Away

Away corresponds to the situation where no face is detected and the difference with a reference image stays below a certain threshold. At this level, no image is transmitted to the other instances. The place is represented by the last image transmitted at the *available* level and clips recorded at the *engaged* one. These images are displayed in a small size on the periphery of the screen and slowly drift in perspective over time.

Clips are displayed as grayscale images. They are normally represented by a single image, the first one, but get promoted to the focus area from time to time to be played at a larger scale if none of the places is at the *engaged* level. The last *available* image degrades over time to make it clear it is not live and mitigate privacy concerns. As illustrated by Figure 3, the filter produces an oil painting effect that rapidly removes details without suppressing all visible information.

Figure 3: Image degradation over time (one minute, two minutes).

The small size of the images displayed at this level invites users who want to see them to move close to the display. If they come close enough, their own Pêle-Mêle will switch to the *engaged* mode and start recording them. We anticipate that this will in turn support the asynchronous creation of common knowledge by reciprocal exchange of video clips (e.g. I saw you watching me opening the present you sent).

3.2 Available

The Pêle-Mêle switches from *away* to this level when it detects a face or an important change between the current scene and the reference image, in which case this reference is also updated. The assumption we make is that such a change is probably caused by incoming people or previously undetected ones. An auditory feedback is generated and the size of the image representing the place on the periphery is slowly increased to a medium size (Figure 4). If a face is found close enough or if significant motion is detected, the Pêle-Mêle switches further to the *engaged* level. If no face is found and the scene doesn't change anymore, it falls back to *away*.

The *available* level is the one for which privacy concerns are the greatest, as it corresponds to situations where someone might be seen by the Pêle-Mêle without being actively engaged in a communication. In order to reduce the risk of unintended privacy exposure, we introduce a delay of several seconds between the capture of the images and their display. The images, however, are immediately processed by the activity detection algorithm. This allows users to prevent the public display of a particular situation by moving out of the camera's field of view to trigger a transition back to the *away* level before the delay expires. As explained above, the last image transmitted will also be rapidly degraded when used at the *away* level to provide some information without unnecessarily exposing privacy.

In a way similar to what Hudson & Smith or Gutwin proposed [2, 1], the delayed video stream is temporally composed to provide awareness of recent activity. Selected past images are alpha-blended with the current one before it is displayed. The alpha value of each image is inversely proportional to its age, which makes it easy to perceive their temporal order (Figure 5). This technique tends to produce composited images with a low contrast, but histogram stretching techniques can be used to alleviate this problem [8]. The selection of past images is a more complex problem. Our current implementation selects a new image every two seconds and uses the last four ones. But these images, like randomly-selected ones, are often void of interesting content. Video summarization techniques could be useful, but they are usually designed for scenarised videos created from multiple sources and associated to an audio channel, while the image streams we process are taken from a unique and fixed viewpoint.

Figure 5: Examples of time composed pictures.

3.3 Engaged

During the transition between the *available* and *engaged* levels, the size of the video stream slowly increases while it moves toward the center of the screen. Auditory feedback accompanies the transition and the delay is progressively suppressed. To achieve this, the stream is accelerated by dropping some of its images in order to catch up with the present. This technique degrades the visual-temporal information in two ways: it deforms motion but also suppresses intermediary frames containing motion-related information. Lossless acceleration techniques (e.g. frame interpolation) were not used due to their high computational cost.

When the transition is finished, live images are displayed in a big size in the focus area and recorded for later use at the *away* and *available* levels. The images of all the places at the *engaged* level are actually alpha-blended together (Fig-

Figure 4: Image size growth during the transition from *away* to *available*.

ure 6). Although the combined display of local and remote participants is known to improve the co-presence feeling [6], the blending of multiple video sources can be quite confusing, e.g. making it difficult to associate faces and backgrounds. To minimize this problem, Pêle-Mêle uses a lower alpha value for local images. This is the only exception to our WYSIWIS design principle.

Figure 6: Two then three users engaged together.

The transition back from *engaged* to *available* triggers when a close face and a significant motion are not detected anymore. During this transition, the size of the video stream decreases while it moves back to the periphery. The stream also temporarily decelerates to introduce the few seconds delay mentioned previously.

4. SUMMARY AND FUTURE WORK

Existing video communication systems lack the ability to move from loosely-coupled to highly-coupled interactions, from casual asynchronous awareness to synchronous face-to-face communication. Pêle-Mêle is our first attempt at addressing this problem using the notions of *variable degree of engagement* and *smooth transitions between degrees*. In this note, we presented the general concept of this system and briefly described the computer vision techniques, the screen layout and the image filtering techniques it uses.

Informal testing indicates that users quickly perceive the three levels of engagement currently supported by the system. The layout, the auditory feedback and the animations help them perceive the transitions, and they quickly understand how these transitions can be triggered by simple movements. The effectiveness of the spatial and temporal filters in mitigating privacy concerns and supporting better awareness over time is more difficult to assess. Long term use and participatory (re)design workshops should help us get user feedback on these important issues and improve the system in the future.

Informal testing already showed that users sometimes misunderstand the delayed display of images at the *available* level as system malfunctions. Future work will address this concern by exploring ways of expliciting the delay, by altering the images or enriching them with abstract visual representations. Future work will also investigate different image-based representations of past activity. We are already investigating new ways of selecting past images and composing them to better support awareness of recent activity at the *available* level. Finally, future work will also seek to support additional degrees of engagement. As an example, we are thinking of using a VoIP application to enrich the current *engaged* level with an audio link when users superimpose their faces.

5. ACKNOWLEDGMENTS

This work has been supported by France Télécom R&D as part of the DISCODOM project. Thanks to Danielle Lottridge for her helpful comments on an earlier version of this note.

6. REFERENCES

[1] C. Gutwin. Traces: Visualizing the Immediate Past to Support Group Interaction. In *Proc. of Graphics Interface*, pages 43–50, May 2002.

[2] S. E. Hudson and I. Smith. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proc. of CSCW'96*, pages 248–257. ACM Press, Nov. 1996.

[3] H. Hutchinson, W. Mackay, B. Westerlund, B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, B. Eiderbäck, S. Lindquist, and Y. Sundblad. Technology probes: Inspiring design for and with families. In *Proc. of CHI 2003*, pages 17–24. ACM Press, Apr. 2003.

[4] W. Mackay. Media Spaces: Environments for Informal Multimedia Interaction. In M. Beaudouin-Lafon, editor, *Computer-Supported Co-operative Work, Trends in Software Series*. John Wiley & Sons Ltd, 1999.

[5] G. McEwan and S. Greenberg. Supporting social worlds with the community bar. In *Proc. of GROUP'05*, pages 21–30. ACM Press, 2005.

[6] O. Morikawa and T. Maesako. HyperMirror: toward pleasant-to-use video mediated communication system. In *Proc. of CSCW'98*, pages 149–158. ACM Press, 1998.

[7] N. Roussel, H. Evans, and H. Hansen. Proximity as an interface for video communication. *IEEE Multimedia*, 11(3):12–16, July-September 2004.

[8] F. Vernier, C. Lachenal, L. Nigay, and J. Coutaz. Interface augmentée par effet miroir. In *Proc. of IHM'99*, pages 158–165. Cépaduès, Nov. 1999.

Beyond "Beyond Being There": Towards Multiscale Communication Systems

Nicolas Roussel
LRI (Univ. Paris-Sud – CNRS), INRIA
Bâtiment 490, Université Paris-Sud
91405 Orsay Cedex, France
roussel@lri.fr

Sofiane Gueddana
LRI (Univ. Paris-Sud – CNRS), INRIA
Bâtiment 490, Université Paris-Sud
91405 Orsay Cedex, France
gueddana@lri.fr

ABSTRACT

Forty years after AT&T's Picturephone, video is still mainly considered as a way to enhance audio communication in an attempt to reproduce face-to-face conditions. In a 1992 paper, Hollan and Stornetta argued that we should develop communication tools that go *beyond being there*. In this paper, we discuss two different interpretations of their analysis. We then propose the concept of *multiscale communication* as an alternative approach for motivating telecommunication research, an approach that aims at creating systems that support a variable degree of engagement, smooth transitions between degrees and integration with other media. Finally, we present three video systems from which the multiscale communication concept emerged and that partly illustrate it.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Computer conferencing, teleconferencing, and videoconferencing; H.1.2 [**Models & Principles**]: User/Machine Systems - Human factors; H.5.2 [**User Interfaces**]: User-centered design; H.5.3 [**Group and Organization Interfaces**]: Collaborative computing

General Terms

Design, Human factors

Keywords

Video-mediated communication, computer-mediated communication, multiscale communication, coordination, communication, collaboration

1. INTRODUCTION

Forty years after AT&T's Picturephone [29], video is still mainly considered as a way to enhance audio communication in an attempt to reproduce face-to-face conditions. Despite what futurologists predicted, videoconferencing has not replaced physical business travel. And although videoconferencing applications are available for free on the most popular software platforms (Microsoft Windows, Linux and Apple Mac OS X), few people actually use them on a regular basis. Oral and text-based communications, like email or instant messaging, remain by far the most popular solutions for asynchronous or distant communication.

Historians and CSCW researchers have investigated the reasons for the failure of traditional videoconferencing (e.g. [29, 12]) and proposed innovative uses of video for mediated communication (e.g. [4, 24]). This research somehow culminated in 1997 with the book *Video-mediated communication* edited by Finn, Sellen and Wilbur [13]. But strangely enough, the interest for innovative uses of video dropped off just as digital media and fast large area networks were becoming ubiquitous. As partly prophesied by Karam [40], the information superhighways killed most of the existing projects, based on analog media, like the US Interstate system killed Route 66:

> "People were not so likely to seek their fortune on the edge of a doomed road, and of those who were already there, fewer and fewer saw any value in upgrading or expanding or - sometimes - doing basic maintenance. After 1956, Route 66 remained important, but its importance was slowly moving away from the concrete toward the glorification of what the highway had been." (S.C. Kelly in *Route 66 - The highway and its people*, cited in [40])

Advances in media and networking technologies have made the implementation of video communication systems considerably easier. DSL technology brings to every home the bandwidth equivalent of a T-2 line, which AT&T used in the early 1970's to carry Picturephone signals. New video codecs such as H.264 promise "ultra-efficient, unprecedented video quality" [2]. But, as far as video-mediated communication (VMC) is concerned, these technologies are only used to create ultra-efficient Picturephones.

The original Picturephone was largely built on the assumption that the addition of sight to sound was both desirable and inevitable [29]. Although this assumption proved to be at least partly incorrect, few people question the motivations of current VMC research: what are we trying to achieve, why are we using video and how does this relate to other communication systems? In a quite influential paper from 1992, Hollan and Stornetta argued that rather than trying to imitate physical proximity, telecommunication research should develop tools that go *beyond being there* [20]. In this paper, we too question the goal of video-mediated communication and telecommunication research in general.

The paper is organized as follows. The next section discusses two different interpretations of Hollan and Stornetta's analysis. We then propose the concept of *multiscale communication* as an alternative approach for motivating telecommunication research, an approach that aims at creating systems that support a variable de-

MM'07, September 23–28, 2007, Augsburg, Bavaria, Germany.

gree of engagement, smooth transitions between degrees and integration with other media. Finally, we present three video systems from which the multiscale communication concept emerged and that partly illustrate it.

2. BEYOND BEING THERE

Being there is of course literally impossible. The expression refers to the concept of *presence*, which Lombard and Ditton define as "the perceptual illusion of nonmediation" [30]. *Being there* also refers to what has long been the main goal of VMC research: "achieving the level of information richness that we currently have in face-to-face interactions" to "interact with others that are far away just as we do with those that are near" [20].

The sense of presence, as defined by Lombard and Ditton, varies according to the media used. *Social presence* [45] and *media richness* [9] theories have been proposed and refined to characterize media, compare them and help people find the ones that maximizes efficiency or satisfaction for a particular task. Much of the research derived from these theories builds on the assumption that increased richness is linked to increased social presence [10]. As an example, the ability to support visual cues such as face expressions, eye contact, gestures or proximity is often said to increase the perceived sense of presence [45], i.e. to decrease the sense of mediation.

In their CHI 1992 paper [20], Hollan and Stornetta question the fundamental goal of telecommunication research. They suggest that instead of trying to imitate face-to-face communication, we should design tools that go *beyond being there*. The conclusion of their paper says:

> "If we ever hope to solve the telecommunication problem, we must develop tools that people would prefer to use even when they have the option of interacting in physical proximity as they have heretofore. To do that requires tools that go *beyond being there*. To create such tools, we suggest framing the problem in terms of needs, media, and mechanisms. The goal then becomes identifying needs which are not ideally met in the medium of physical proximity, and evolving mechanisms which leverage the strengths of the new medium to meet those needs."

This analysis has been quite popular and has inspired a number of systems. However, a broad look at these systems shows two very different interpretations, corresponding to different meanings of the word beyond: *greater than* and *other than*.

2.1 "Beyond" as "greater than": the ultra-high fidelity approach

Hollan and Stornetta ask the following question: "what would happen if we were to develop communication tools with a higher information richness than face-to-face?". Some people – notably from the Multimedia research community – take this as an invitation to pursue the prevailing technology-driven approach to improve existing systems without questioning them. From this perspective, technical limitations still explain the relative failure of video-mediated communication, and further technical developments will help solve the remaining issues:

> "Systems rarely support more than two participating sites, and specially equipped rooms are often required. Frame rates and image quality lag expectations, and the resulting experience is of blurry television watching rather than personal interchange. Our intention in Coliseum has been to push the envelope in all dimensions of this technology – display frame rate and resolution, response latency, communication sensitivity, supported modalities, etc." [3]

> "Why have current alternatives to physical travel such as video conferencing technology not replaced even more business travel? One hypothesis is that it is because such technology is not *immersive*." [26]

> "New sensors (e.g., touch, smell, taste, motion, etc.) and output devices (e.g., large immersive displays and personal displays integrated with eye glasses) offer the opportunity for more intimate and sensitive interaction with a remote environment. And, continued development of semiconductor technology will bring real-time three-dimensional virtual environments to every computing and communication platform. As one participant said, *interacting with a remote environment should be better than being there*." [43]

This approach focuses on *immersive*, *experiential* and *effective* telepresence[1], the proclaimed goal being to make the communication more *natural*, more *intuitive* and more *realistic*. Recent publications have indeed demonstrated impressive progress toward multiple viewpoints systems and immersive displays (e.g. blue-c [17], Twister [48], BiReality [26], Coliseum [3], MultiView [35]). But this approach has several problems. First, it focuses on media and mechanisms but often neglects user needs. Second, in order to "beat the physical proximity", it pursues the same immediate goal of imitating it. The mark is just set higher than before, high-fidelity sight and sound being considered as minimum requirements to be complemented with new technologies. Lastly, these new technologies often create their own problems, resulting in an endless quest for performance and fidelity:

> "Realistically, there are numerous developments that remain before this could be considered a viable alternative to travel for collaborative remote conferencing. Obvious improvements include increasing the frame rate, reducing latency, raising the quality at which people are displayed, and reconfiguring computation to enable more advanced features (such as head tracking)." [3]

> "Probably the biggest negative comment from users concerns the latency of the current system. One-way latency of the video is almost 700ms, so it is very noticeable. (...) We hope that the next generation of video compression cards will have reduced latency." [26]

The ultra-high fidelity approach will hopefully lead to efficient lifelike conferencing systems. These systems might even provide services that remain valuable in the case of physical proximity, such as the ability to simultaneously manipulate shared artifacts. But their focus on synchronous face-to-face communication, combined with complex hardware and software requirements, will limit their use to formal, planned and highly engaged interactions.

[1]These three terms were used for a series of workshops associated to the ACM Multimedia conference in 2002, 2003 and 2004.

2.2 "Beyond" as "other than": the high-diversity approach

Formal interactions account for only part of typical group activity. Various studies have demonstrated the importance of more spontaneous, opportunistic, informal interactions [23]. Studies of co-located interactions have also shown the crucial role of visual information in monitoring and tracking availability among coworkers [49], which makes video an interesting technology for asynchronous or remote collaboration. Indeed, tracking the availability of other people for unscheduled communication is a typical need not ideally met in the physical world: how many visits to a colleague's office do you need to make before you find him or her available for discussion?

Mediaspace studies [4, 31] have investigated the potential uses of video to support collaborative activities ranging from casual awareness and informal talks – side-by-side interactions – to formal focused face-to-face communications. A variety of new services have been proposed. As an example, in addition to traditional videoconferencing, the RAVE mediaspace [15] made the following ones available: *background* (a view of a public area, used as the default connection), *glance* (a short one-way video connection), *sweep* (a series of glances), *office share* (a long-term audio and video link). These synchronous analog services were also complemented by the Portholes system [11] that presented regularly updated digitized images on the workstation screen.

While the ultra-high fidelity approach focuses on the foreground activity made possible by physical proximity, most mediaspace studies were interested in the background and possibly unconscious forms of communication that go with it. One interesting finding, for example, is that in order to use it for background communication, one might need to reduce the information transmitted on a particular channel: Riesenbach [39] explains how lowering the resolution and frame rate of the permanent video connections of the Ontario Telepresence Project made them more socially acceptable by reducing the attention of the recipient and preserving the privacy of the sender.

A number of other techniques have been proposed to help mediaspace users find the appropriate trade-off between awareness and privacy, including notification and control mechanisms [15], image and sound filtering [46, 50] and synthetic presentation of presence information [21]. Researchers later explored even more abstract, subtle and implicit forms of communication through lights, haptics and scent by taking advantage of a particular context[2], such as the intimate relation between two people [47, 5, 6]. But the most interesting aspect of mediaspace studies, we believe, is that they promoted the idea that a gradual engagement in communication is desirable and demonstrated that it is possible. In the next section, we will explain how this notion can be expanded to move on towards a new generation of communication systems.

3. TOWARDS MULTISCALE COMMUNICATION SYSTEMS

Although everyone seems to agree that we should develop systems that go *beyond being there*, not everyone seems to agree where to go. An ultra-high fidelity interpretation of Hollan and Stornetta's analysis drives a number of researchers to a potentially endless quest for improving existing conferencing services without questioning their goal. A more social approach, exemplified by mediaspace studies, reconsiders the problem of video-mediated communication and proposes an increasing number of alternative services to traditional conferencing. But how do these services relate one to another? How do they relate to the many communication systems we already use, like email, instant messaging or the telephone? Can we structure their design space in a way that includes both high fidelity systems for face-to-face interactions as well as subtle, implicit and abstract forms of communications?

Gaver et al. proposed the *degree of engagement* and the *amount of planning* as two dimensions to analyze collaborative work [15]. The RAVE services (*background*, *sweep*, *glance*, *office share* and *vphone*) reflected this idea of having multiple degrees of engagement. Although less interested in the amount of planning, we believe the notion of selective engagement is an important one that can help structure the design space of communication systems. We also believe this notion could help users better choose the right communication service for a particular context.

Gaver et al. had a quite simple definition for the degree of engagement: "the extent to which a shared focus is involved". Other researchers have developed similar – although more refined – concepts. Fish et al. [14], for example, talked about the necessary balance between *accessibility* (access to others), *privacy* (control over the available information about oneself) and *solitude* (control over others' intrusion in one's space and time). Greenhalgh and Benford [16] also suggested that users should be able to separately control their *nimbus* (one's manifestation or observability) and *focus* (one's allocation of attention). This idea was notably applied to video communication inside a Collaborative Virtual Environment [38] and more recently in the Community Bar awareness system [32]. Based on these different concepts, our own definition for the degree of engagement is the following: "the extent to which users are ready to expose themselves and open to others".

The Community Bar presence item proposes six degrees of engagement based on combinations of the following attributes: a two-state color activity indicator, the user name, a status message, a static picture, a webcam snapshot and a fast frame rate video connection. Sliders make it possible to control one's focus on each of the other users. A nimbus slider also makes it possible to specify a level of detail which others can only see up to, but not beyond (using their focus slider). Although this system makes use of video, this use is quite limited. One reason for this is probably that the Community Bar presence item, as the name suggests, is a tool for presence awareness, not something that aims at supporting the full range of collaborative activities.

Previous research on video-mediated communication has demonstrated that video, through its different forms, can be used to support a wide range of activities. Mediaspaces are probably the closest attempt at creating a single system to support the full range of these activities. We believe this should be the goal of future VMC research and development. This goal is not new. It was one of RAVE designers' for example. But it seems to have been abandoned on the way. The following problems, in particular, should be explored:

- Beyond[3] snapshots and full-rate: How can we use video to implement degrees of engagement other than static pictures and high-quality streams? How many degrees can we create? Can we create a continuum of degrees?

[2]The idea that taking a particular context into account can help reduce a message while preserving its general meaning is not new. According to [36], Victor Hugo was on vacation when his book *Les Misérables* was published. Curious to know how it was doing, he sent a telegram to his publisher, reading simply "?". The publisher replied in an equally short way: "!".

[3]Use of the word beyond is not coincidental. As we have seen, it leaves some space for reader interpretation…

- Beyond buttons, sliders and labels: How can we move from one degree to another? How can we perceive a remote person's degree? How can we negotiate degrees with remote people? Can we avoid explicit dialog boxes and support more intuitive interactions?
- Beyond video: How can we combine video with other media? (e.g. email, the telephone, instant messaging, the Web)

As illustrated by the case of the permanent connections of the Ontario Telepresence Project, the level of detail of an image stream is probably related to the associated degree of engagement: the bigger, the more colorful, the sharper and the more frequent the images are, the more they expose the person they show and will probably attract the attention of the person that sees them. In addition to these attributes, other characteristics of an image stream could be manipulated to alter the associated engagement degree. As illustrated by Figure 1, filtering techniques can be used to degrade images [50] while temporal compositions can provide awareness of past activity [21, 19]. One could certainly imagine other image filtering techniques to enrich the video as well as temporal techniques to degrade it (e.g. by introducing a controlled delay). More subtle filters could also eliminate some details while enhancing others [28, 7].

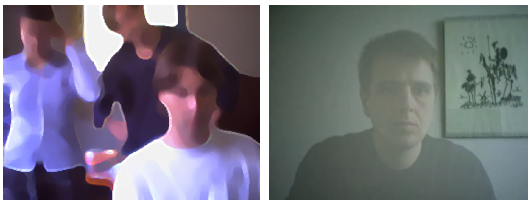


Figure 1: Degrading (left) or enriching (right) a video stream.

Transitions between engagement degrees pose two kinds of problems. First, new interaction techniques are required to specify the desired degree. These techniques need to be as direct and concise as possible since managing one's communications should not become a primary activity itself. The camera, in this context, is probably an interesting input device and other sensors might also be useful. Feedback mechanisms such as animations can probably help make the user aware of the transitions initiated by remote partners. Combining the video system with other communication tools again requires the design of appropriate interaction techniques and feedback mechanisms. As an example, one might want to temporarily use a mobile phone as an additional audio channel to an existing video communication. Combining synchronous and asynchronous communication also poses some interesting problems.

To summarize:

- We believe our goal should be to develop new communication systems that support a variable degree of engagement.
- These systems should support smooth transitions between degrees. They should also support smooth integration with other media or communication systems.
- Video is a good starting point, as it has already been shown to support a wide range of collaborative activities and can also be used as an input channel for Human-Computer interaction.

A *multiscale world* is defined by Jul and Furnas as a world "in which information can exist at multiple levels of detail" [27]. The degree of engagement, as we see it, somehow corresponds to the level of detail of the communication. Therefore, we propose to use the term *multiscale communication system* to designate a communication system that supports a variable degree of engagement. Smooth transitions between degrees of engagement correspond to smooth variations of the level of detail. In Zoomable User Interface terms [37], we might call them *continuous zooming*. Enriching or degrading a video stream can change both its meaning and level of detail and might thus be considered as the equivalent of a *semantic zoom*.

4. EXAMPLES

We will now present three video systems that partly illustrate the concept of multiscale communication we just introduced. This section will complement previously-published descriptions of these systems by emphasizing aspects of their design and use that are related to the concepts of variable degree of engagement, smooth transitions between degrees and integration with other media.

The first system, VideoServer, shows how focus and nimbus control mechanisms can be used to combine synchronous video services, and how these services can be integrated with asynchronous text-based communication to support lightweight coordination. The second system, VideoProbe, shows how activity sensing techniques can be used to support both implicit (i.e. peripheral) and explicit (i.e. highly engaged) interactions, and transitions from asynchronous to almost synchronous communication. The last system, MirrorSpace, further illustrates the use of sensing techniques to support the implicit control of the degree of engagement in a synchronous communication through the usual body language.

It is important to understand that these systems were not designed according to the multiscale communication approach, but that the concept emerged from them. In other words, these examples are not here to "validate" the concept but rather to explain its genesis.

4.1 VideoServer

VideoServer [41] was designed as a tool to support the creation of a highly tailorable Web-based mediaspace. It is a personal HTTP server that allows a user to make live or pre-recorded images and video streams accessible to other users through simple URLs (Figure 2). In addition to other specific protocols, VideoServer is able to transmit video data to client applications on the HTTP connection itself. In this case, single images are sent as JPEG-compressed data, which can be displayed by any HTML rendering engine in place of an ordinary JPEG image, without any plug-in. Video streams are sent as a server-pushed series of JPEG-compressed images that some HTML renderers can also display in place of an ordinary image[4].

```
http://server/grab/video
http://server/push/video?framerate=5&size=QSIF
http://server/push/video?framerate=25&size=SIF
```

Figure 2: VideoServer URLs requesting a single image, a low frame rate 160x120 video and a high frame rate 320x240 video (all images are captured in real-time).

[4]Gecko, the Mozilla HTML rendering engine is one of them. Mozilla applications such as Camino and Firefox (two Web browsers) or Thunderbird (an email client) can thus display VideoServer streams without any plug-in.

By using URLs such as those of Figure 2, users can easily integrate live images and video streams into email messages (Figure 3, left) and existing or new HTML documents (Figure 3, right). An interesting use of this feature that users developed is to include a live snapshot of one's office in one's email signature or in a Web page that shows your contact information so that people who want to reply to one of your emails or call you can see if you're available for discussion. This ability to provide access to synchronous video services from Web publishing and email, two rather low paced asynchronous media, is a good example of the cross-media integration mentioned in the previous section.

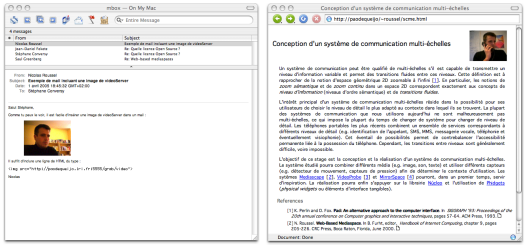


Figure 3: Live VideoServer images displayed in Apple's Mail application and the Camino Web browser. Images are captured and transmitted every time the HTML message or document is rendered by the application.

Awareness views similar to Portholes are easily created by users themselves, by including images from several servers in a single HTML document and using a timer to reload it at regular intervals. Basic image and video services can also be combined to support more complex interactions. A few lines of JavaScript, for example, can turn a static picture into a medium frame-rate video (e.g. 15 fps) when the mouse moves over it and pop up a new window displaying a high frame-rate and resizable stream when one clicks on it (Figure 4). While a previous study suggested that people have difficulty extracting information from snapshots unless the resolution is at least 128x128 pixels [25], experience with this three-scale focus control indicates that snapshot resolution can be reduced up to 80x60 as the ability to turn them into video streams helps resolve ambiguities.

As most mediaspaces and unlike webcam software, VideoServer provides users with notification and access control mechanisms. For every request it receives, it executes an external program (a Python script) with arguments indicating the name of the remote machine, possibly the remote user's login name, the resource that led to the server (the HTTP referrer) and a description of the requested service. The external program uses this contextual information to generate auditory or on-screen notifications (Figure 5) and sends back to the server a description of the service to be executed. This description can be inferred from a set of pre-defined rules or negotiated with the user through some interactive dialog.

An important feature of VideoServer's control mechanism is that the external program is not limited to a binary accept/refuse choice but can freely redefine the service to be executed. It can for example request that a spatial filter be applied on the images, which the remote person will probably notice (Figure 6, image 2). It can redirect the client to another server. But it can also substitute a pre-recorded image or sequence to the live stream. This feature proved particularly useful as it supports the creation of ambiguities and stories [1]. Seeing the third image of Figure 6, for example, one might assume that the remote person is absent. Yet seeing this particular image too often might indicate that she simply doesn't want us to know if she is there. Seeing the fourth image might indicate that she will be away for some time.

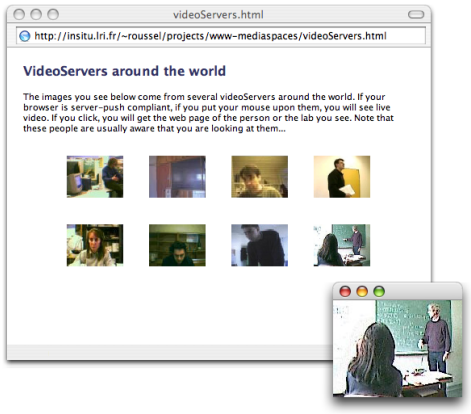


Figure 4: Focus control: from a low resolution snapshot in a Portholes-like awareness view to a high frame rate independent video that the user can freely move and resize.

Figure 5: Sample on-screen notification showing a description of the requested service, the document to contain the requested images and the remote user's address.

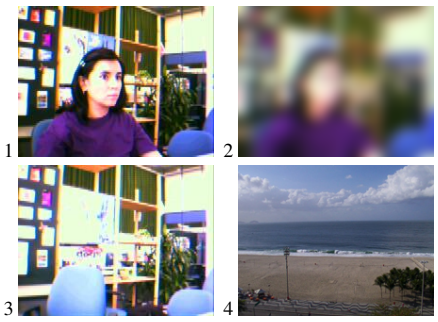


Figure 6: Nimbus control: image captured by the camera (1), filtered image (2), ambiguous pre-recorded image (3) and explicit absence indicator (4).

As we have seen, VideoServer makes it possible to combine synchronous video services with asynchronous communication via email or Web pages. It also provides users with flexible and powerful scripting mechanisms to control their focus and nimbus. Mastering these mechanisms, however, requires some programming knowledge. We will now describe two other systems that illustrate

more direct and intuitive ways of varying one's degree of engagement.

4.2 VideoProbe

VideoProbe [22] was created as part of INTERLIVING[5], a multidisciplinary european project focused on the design of new technologies to support communication among family members located in different households. VideoProbe allows a group of people to share their daily lives by exchanging pictures. It physically consists in a box containing a screen, two speakers and a camera connected to a separate computer, itself connected to the Internet (Figure 7). A specific software analyzes the images captured by the camera in real-time and decides when a picture should be taken and transmitted to similar devices installed in other households (only pictures are exchanged, not video streams).

Figure 7: VideoProbe.

As long as the scene observed by the camera doesn't change, the screen stays blank (Figure 8, image 1). If a change is detected, the software gradually displays the captured images, turning the screen into a mirror (Figure 8, images 2 and 3). If the same observed change persists more than three seconds, a picture is automatically transmitted to the other VideoProbes. A growing translucent rectangle indicates the remaining time (Figure 8, images 4 and 5): when the rectangle reaches the full size of the video frame, an auditory cue is played, the picture is taken, displayed bigger and correctly oriented for three seconds (Figure 8, image 6) and then transmitted to the other VideoProbes. If the scene doesn't change anymore, the screen gradually returns to its blank state. Otherwise, new pictures can be taken and transmitted as just described.

Figure 8: Transitions between the *sleep mode* (1), the *mirror mode* (2 to 5) and the *picture transmission mode* (6).

A remote control allows to switch the system into a browsing mode that shows the pictures taken by all the connected VideoProbes. Within this mode, users can delete selected pictures or save them in a persistent album. Pictures not saved in the album gradually loose their colors and contrast and eventually disappear from the browsing interface after a few days (Figure 9).

[5] http://interliving.kth.se/

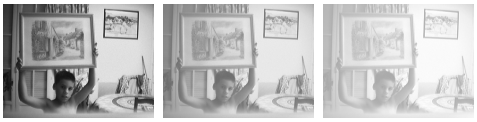


Figure 9: Picture aging in the *browsing mode* (the actual process takes about five days).

As confirmed by long-term user studies [8], the smooth transitions between the different operation modes of VideoProbe play an essential part in making the interaction simple, quick and easy. The combined use of movement detection and delayed picture taking allows to quickly switch the device from an idle state to one where it is ready to communicate while still offering an easy way to back off, as continuous move prevents the system from taking pictures. This was quickly understood by users without formal training and even turned into a little game which goal was to take a picture of an empty room, i.e. move outside the field of view of the camera at the exact moment when the picture was taken (which is in fact particularly hard to achieve).

VideoProbe supports both *explicit* and *implicit* forms of communication. The explicit form takes place when the user is consciously using the system to transmit a particular picture (Figure 10, left). The implicit form typically takes place when someone enters the room and stays there for some reason but does not pay attention to the device (Figure 10, right). In that case, the persistent scene change triggers the taking of a picture and its transmission but the user usually becomes aware of it only when he or she hears the auditory notification.

The implicit form of communication proved very useful for maintaining group awareness as it usually produces pictures that users would not or could not take themselves. At the same time, because of its motion-based control, VideoProbe was perceived as less intrusive and more flexible than a purely time-based approach that would have taken pictures at regular intervals. User motion indirectly determines the rate at which the system transmits images. And although the maximum rate is quite limited (about 10 to 15 frames per second), the system was sometimes used while discussing over the phone as an acceptable replacement for a videoconferencing service. This particular example again illustrates how a single video communication system can support a variable degree of engagement ranging from asynchronous communication to synchronous one.

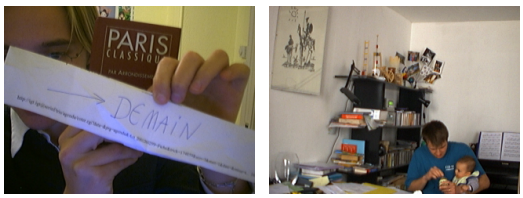


Figure 10: Explicit ("I'll be in Paris tomorrow") and implicit uses of VideoProbe.

The process of picture taking is a slow one during which the presentation of the images captured by the camera is gradually transformed until they reach the state where one will be taken and transmitted: images first fade in and are then gradually covered by the translucent rectangle indicating the remaining time. The grad-

ual degradation of the pictures that have been received follows the same approach: pictures don't disappear suddenly but fade away. As users had the opportunity of canceling the picture taking process, they also have the opportunity to literally save the taken pictures. This shows how the notion of variable engagement can even be used in the case of purely asynchronous communication

Our next example illustrates further the notion of gradual and intuitive engagement in synchronous communication.

4.3 MirrorSpace

MirrorSpace [42] is another video communication system designed for the INTERLIVING project. Whereas existing video systems usually create a shared space corresponding to a particular interpersonal distance, the goal of MirrorSpace was instead to create a continuum of space that would allow a variety of interpersonal relationships to be expressed.

Figure 11: MirrorSpace.

MirrorSpace relies on a mirror metaphor (Figure 11). Live video streams from all places connected through the system are superimposed on a single display on each site so that people see their own reflection combined with the ones of remote persons. In order to support intimate forms of communication where people might want to look into each other's eyes, the camera has been placed right in the middle of the screen. This setup allows users to come very close to the camera while still being able to see the remote people and interact with them. MirrorSpace also includes an ultrasonic proximity sensor that measures the distance to the closest object or person in front of it. A blur filter is applied on the images displayed to visually express a distance computed from the local and remote sensor values. Blurring distant objects and people provides a *social catalyst* [28] to support and encourage distributed interaction. It allows one to perceive their movement or passing with a minimum involvement. It also offers a simple way of initiating or avoiding a change to a more engaged form of communication by simply moving closer (Figure 12) or further away.

Figure 12: Reducing the blur effect by moving closer.

MirrorSpace has been presented to the public in several art exhibitions. In one exhibition, two prototypes were placed inside a 3x3m cubicle that enabled people to directly see and hear each other. In another exhibition, they were completely isolated from each other. In other cases, they were set up in a way that people could hear without being able to see each other directly (e.g. separated by a thin wall or placed back to back). Several hours of video were shot during the exhibitions and later analyzed. Although the context of an art exhibition is somewhat particular, several interesting observations were made that are probably inherent to the system.

Proximity sensing and blur filtration help creating an intimate relationship between users and the system. People like the idea that the system is reacting to them and not just taking images from them, that they are in control and not only the subject. When they see another person appearing next to them on the screen, many people turn over, looking for that person behind them. As previously reported by other studies (e.g. [33]), this shows that the superposition of images creates a strong sense of shared space. The particular placement of the camera, which allows people to come really close to it, turns this shared space into an intimate one. Many people get surprised and even disturbed by this intimacy when a stranger appears too close to them on the screen, but proximity sensing and blur filtration allow them to simply step back to disengage and alter the display.

A recent study showed that blur filtration fails at providing an obfuscation level that could balance privacy and awareness for home situations [34]. Yet, we strongly believe that this type of filtering is still valuable. Not because of what it tries to remove, but because of what it adds: the filter shows the remote people that we don't want them to observe. Of course, there's no guarantee that they won't, but we know that they know they're not supposed to do so. The stronger the filter, the stronger we insist on the fact that it is socially unacceptable for them to observe. Blur filtration can be seen as a way to enrich the video communication to indicate the desire for a lesser-engaged form of communication. The fact that it does not necessarily enforce this lighter form of communication leaves room for negotiation between people.

In MirrorSpace, the strength of the blur effect applied on an image is computed from the proximity sensor values of all the connected devices. In the simplest case, the strength is the result of a transfer function applied to the local sensor value. The transfer function makes it possible to adapt the system to the particular geometry of the room where it has been installed. A more interesting case is when the blur effect applied on the image of a remote person is computed from both the local and remote sensor values. Using the sum of these values, for example, makes it possible for two people, Chris and Steve for example, to negotiate a common degree of engagement:

- If Chris moves closer to the device, the image of Steve on his screen and his own image on Steve's screen will get sharper
- Steven will then be able to accept the new engagement degree, to increase it further by also moving closer to the device or to go back to the previous state by stepping back

This example shows that it is possible to create communication systems that uses at least part of the physical body language to negotiate a common engagement degree in a way similar to what had been proposed by Greenhalgh and Benford for virtual environments [16].

5. CONCLUSION

In this paper, we have introduced the concept of *multiscale communication* as an alternative approach for motivating video-mediated communication research, and telecommunication research in general. This approach aims at creating systems that support a variable degree of engagement, smooth transitions between degrees and integration with other media. We have also presented the three video systems from which this concept originated, each

of them illustrating one or more aspects of it (e.g. integration with other media, transitions between asynchronous and synchronous communication, intuitive control of the engagement degree).

We are currently working on a series of new communication systems to further explore the design space offered by the multiscale approach. The first one, Pêle-Mêle [18], is another multiparty video system that combines computer vision techniques, spatial and temporal filtering and an original layout to support both asynchronous and synchronous communication, three degrees of engagement and the transitions between them. But the multiscale approach to communication is not limited to video. We are particularly interested, for example, in the potential transitions between various forms of communication based on text (e.g. email and instant messaging), audio, video and shared artifacts.

We hope the multiscale approach will stimulate other researchers interested in multimedia communication. We are particularly curious about the other parallels that might be found between Computer-Mediated Communication and Information Visualization. As an example, Shneiderman's visual information seeking mantra seems particularly relevant to the way we usually engage in a communication with another person, and summarizes pretty well the idea of gradual engagement:

> "Overview first, zoom and filter, then details-on-demand" [44]

After all, isn't communication the process of exchanging information?

6. ACKNOWLEDGEMENTS

This work has been supported by France Télécom R&D as part of the DISCODOM projects (2005-2008). Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen and Wendy Mackay all contributed to the design and implementation of the three systems we described. Thanks to Rosane Ushirobira, Stéphane Chatty, Stéphane Sire and Angela Sutan for their helpful comments on an earlier version of this paper.

7. REFERENCES

[1] P. M. Aoki and A. Woodruff. Making space for stories: ambiguity in the design of personal communication systems. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 181–190, New York, NY, USA, 2005. ACM Press.

[2] Apple. QuickTime and MPEG-4: Now Featuring H.264. Technology brief, Apple, Apr. 2005.

[3] H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender. Understanding performance in Coliseum, an immersive videoconferencing system. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(2):190–210, May 2005.

[4] S. Bly, S. Harrison, and S. Irwin. Mediaspaces: Bringing people together in a video, audio and computing environment. *Commun. ACM*, 36(1):28–47, Jan. 1993.

[5] S. Brave, H. Ishii, and A. Dahley. Tangible interfaces for remote collaboration and communication. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 169–178. ACM Press, 1998.

[6] A. Chang, B. Resner, B. Koerner, X. Wang, and H. Ishii. Lumitouch: an emotional communication device. In *CHI'2001 extended abstracts on Human factors in computer systems*, pages 313–314. ACM Press, 2001.

[7] D. J. Chatting, J. S. Galpin, and J. S. Donath. Presence and portrayal: video for casual home dialogues. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 395–401, New York, NY, USA, 2006. ACM Press.

[8] S. Conversy, W. Mackay, M. Beaudouin-Lafon, and N. Roussel. VideoProbe: Sharing Pictures of Everyday Life. Rapport de Recherche 1409, LRI, Université Paris-Sud, France, Apr. 2005. 8 pages.

[9] R. Daft and R. Lengel. Information richness: a new approach to managerial behavior and organizational design. In L. Cummings and B. Staw, editors, *Research in organizational behavior 6*, pages 191–233. JAI Press, 1984.

[10] A. R. Dennis and J. S. Valacich. Rethinking media richness: Towards a theory of media synchronicity. In *HICSS '99: Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*, page 1017. IEEE Computer Society, 1999.

[11] P. Dourish and S. Bly. Portholes: Supporting Awareness in a Distributed Work Group. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 541–547. ACM Press, 1992.

[12] C. Egido. Videoconferencing as a Technology to Support Group Work: A Review of its Failure. In *Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work*, pages 13–24. ACM Press, Sept. 1988.

[13] K. Finn, A. Sellen, and S. Wilbur, editors. *Video-Mediated Communication*. Lawrence Erlbaum, Apr. 1997. 584 pages.

[14] R. S. Fish, R. E. Kraut, R. W. Root, and R. E. Rice. Video as a technology for informal communication. *Communications of the ACM*, 36(1):48–61, 1993.

[15] W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter, and W. Buxton. Realizing a Video Environment: EuroPARC's RAVE System. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 27–35. ACM Press, 1992.

[16] C. Greenhalgh and S. Benford. Massive: a collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3):239–261, 1995.

[17] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, and O. Staadt. blue-c: A spatially immersive display and 3D video portal for telepresence. *ACM Transactions on Graphics, Proceedings of SIGGRAPH 2000*, 22(3):819–827, 2003.

[18] S. Gueddana and N. Roussel. Pêle-mêle, a video communication system supporting a variable degree of engagement. In *Proceedings of ACM CSCW'06 Conference on Computer-Supported Cooperative Work*, pages 423–426. ACM Press, Nov. 2006.

[19] C. Gutwin. Traces: Visualizing the Immediate Past to Support Group Interaction. In *Proceedings of Graphics Interface*, pages 43–50, May 2002.

[20] J. Hollan and S. Stornetta. Beyond being there. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 119–125. ACM Press, 1992.

[21] S. E. Hudson and I. Smith. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proceedings of ACM CSCW'96 Conference on Computer-Supported Cooperative Work, Boston, Mass.*, pages 248–257. ACM Press, Nov. 1996.

[22] H. Hutchinson, W. Mackay, B. Westerlund, B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, B. Eiderbäck, S. Lindquist, and Y. Sundblad. Technology probes: Inspiring design for and with families. In *Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems*, volume 5(1) of *CHI Letters*, pages 17–24. ACM Press, Apr. 2003.

[23] E. Isaacs, S. Whittaker, D. Frohlich, and B. O'Conaill. Informal communication re-examined: New functions for video in supporting opportunistic encounters. In K. Finn, A. Sellen, and S. Wilbur, editors, *Video-mediated communication*. Lawrence Erlbaum Associates, 1997.

[24] H. Ishii. Integration of Shared Workspace and Interpersonnal Space for Remote Collaboration. In M. Beaudouin-Lafon, editor, *Computer-Supported Co-operative Work, Trends in Software Series*. John Wiley & Sons Ltd, 1999.

[25] B. Johnson and S. Greenberg. Judging People's Availability for Interaction from Video Snapshots. In *Proceedings of the Hawaii International Conference on System Sciences, Distributed Group Support Systems Minitrack*. IEEE Press, Jan. 1999.

[26] N. Jouppi, S. Iyer, S. Thomas, and A. Slayden. Bireality: mutually-immersive telepresence. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 860–867. ACM Press, 2004.

[27] S. Jul and G. W. Furnas. Critical zones in desert fog: aids to multiscale navigation. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 97–106, New York, NY, USA, 1998. ACM Press.

[28] K. Karahalios and J. Donath. Telemurals: linking remote spaces with social catalysts. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 615–622, New York, NY, USA, 2004. ACM Press.

[29] K. Lipartito. PicturePhone and the Information Age: The Social Meaning of Failure. *Technology and Culture*, 44(1):50–81, Jan. 2003.

[30] M. Lombard and T. Ditton. At the heart of it all: The concept of presence. *Journal of Computer-Mediated Communication*, 3(2), Sept. 1997.

[31] W. Mackay. Media Spaces: Environments for Informal Multimedia Interaction. In M. Beaudouin-Lafon, editor, *Computer-Supported Co-operative Work, Trends in Software Series*. John Wiley & Sons Ltd, 1999.

[32] G. McEwan and S. Greenberg. Supporting social worlds with the community bar. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 21–30, New York, NY, USA, 2005. ACM Press.

[33] O. Morikawa and T. Maesako. HyperMirror: toward pleasant-to-use video mediated communication system. In *Proceeding of ACM CSCW'98 Conference on Computer-Supported Cooperative Work, Seattle, Mass.*, pages 149–158. ACM Press, 1998.

[34] C. Neustaedter, S. Greenberg, and M. Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction*, 13(1):1–36, 2006.

[35] D. Nguyen and J. Canny. Multiview: spatially faithful group video conferencing. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 799–808, New York, NY, USA, 2005. ACM Press.

[36] T. Nørretranders. *The user illusion: cutting consciousness down to size*. Penguin books, 1991.

[37] K. Perlin and D. Fox. Pad: An alternative approach to the computer interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 57–64. ACM Press, 1993.

[38] G. Reynard, S. Benford, C. Greenhalgh, and C. Heath. Awareness driven video quality of service in collaborative virtual environments. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 464–471, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[39] R. Riesenbach. Less is more (more or less...). White Paper, Telepresence Systems, 1996.

[40] R. Riesenbach, W. Buxton, G. Karam, and G. Moore. Ontario Telepresence Project. Final report, Information technology research centre, Telecommunications research institute of Ontario, Mar. 1995.

[41] N. Roussel. Mediascape: a Web-based Mediaspace. *IEEE Multimedia*, 6(2):64–74, April-June 1999.

[42] N. Roussel, H. Evans, and H. Hansen. Proximity as an interface for video communication. *IEEE Multimedia*, 11(3):12–16, July-September 2004.

[43] L. A. Rowe and R. Jain. ACM SIGMM retreat report on future directions in multimedia research. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(1):3–13, 2005.

[44] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society, 1996.

[45] J. Short, E. Williams, and B. Christie. *The Social Psychology of Telecommunications*. Wiley, New York, 1976.

[46] I. Smith and S. Hudson. Low disturbance audio for awareness and privacy in media space applications. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 91–97. ACM Press, 1995.

[47] R. Strong and B. Gaver. Feather, scent and shaker: supporting simple intimacy. In *Proceedings of ACM CSCW'96 Conference on Computer-Supported Cooperative Work*, pages 29–30. ACM Press, Nov. 1996.

[48] K. Tanaka, J. Hayashi, M. Inami, and S. Tachi. Twister: An immersive autostereoscopic display. In *VR '04: Proceedings of the IEEE Virtual Reality 2004 (VR'04)*, pages 59–66, Washington, DC, USA, 2004. IEEE Computer Society.

[49] S. Whittaker. Video as a technology for interpersonal communications: a new perspective. In A. Rodriguez and J. Maitan, editors, *Proceedings of Multimedia Computing and Networking 1995*, volume 2417, pages 294–304. SPIE, 1995.

[50] Q. Zhao and J. Stasko. Evaluating Image Filtering Based Techniques in Media Space Applications. In *Proceeding of ACM CSCW'98 Conference on Computer-Supported Cooperative Work, Seattle, Mass.*, pages 11–18. ACM Press, 1998.

Ametista: a mini-toolkit for exploring new window management techniques

Nicolas Roussel
Laboratoire de Recherche en Informatique (LRI) & INRIA Futurs*
Bât 490, Université Paris-Sud
91405 Orsay Cedex, France
+33 1 69 15 66 23
roussel@lri.fr

ABSTRACT
Although the HCI research community has contributed a number of metaphors, interaction techniques and layout algorithms to improve window management tasks, most of these ended as prototypes and only a few were implemented in real window managers. In this paper, we present Ametista, a mini-toolkit designed to facilitate the exploration of new window management techniques using both low-fidelity prototyping and a high-fidelity approach based on X Window application redirection.

Keywords
Window management, graphical interaction, prototyping, application redirection, OpenGL, X Window system, VNC

INTRODUCTION
In [16], Myers defines a window manager as "*a software package that helps the user monitor and control different contexts by separating them physically onto different parts of one or more display screens*". He adds: "*Before window managers, people had to remember their various activities and how to switch back and forth*". Twenty years after the general adoption of the desktop metaphor [12] and overlapping windows [5], the growing range of activities supported by interactive computer applications has brought us back to the point where it is again difficult to remember these activities and organize them.

Over the past few years, a number of novel metaphors, interaction techniques or layout algorithms have been proposed to extend or replace the desktop metaphor such as the *pile* metaphor [14], *elastic windows* [13], *tabbed, rotated and peeled back windows* [2] or constraint-based layout [1]. However, most of these proposals were never implemented in a 'real' window manager. *Piles*, for example, were prototyped with Macromind Director; *elastic windows* were implemented within custom applications and *rotated and peeled back windows* were prototyped in Tcl/Tk.

With the advent of hardware-accelerated graphics, the graphics libraries available to application developers have tremendously improved in recent years. Performance of graphics hardware is increasing faster than Moore's law, supporting more and more advanced graphics functions. The Direct3D and OpenGL libraries, for example, natively support arbitrary 3D transformations, double buffering, Z-buffering, alpha blending, texture mapping and material lighting. As illustrated by [3], all these graphics functions make it possible to efficiently implement advanced graphical interaction techniques such as *toolglasses* and *magic lenses* [4] or *zoomable interfaces* [18].

By contrast, the graphical libraries used for window management have not followed this trend, making it difficult or impossible to use texture mapping, alpha blending or arbitrary geometric transformations at the level of windows. Until recently, the three most popular windowing systems were still based on graphics libraries designed in the 1980s: GDI for Microsoft Windows, QuickDraw for Apple Mac OS and the Xlib for the X Window system. The graphics models associated to these libraries were relatively simple. In particular, the color of each pixel on the screen was determined by a single application through a few logical operations (e.g. *and*, *or*, *xor*) applied on elementary 2D primitives. The main reason why these models were so simple is that the hardware available when they were designed was barely powerful enough for them. It actually took several years before GDI, QuickDraw or X server implementations could take advantage of hardware acceleration provided by consumer-level graphics hardware.

We believe that the large difference between the graphics models available to applications and window managers is one of the reasons why many innovative graphical interaction techniques were never taken to the point where they can be used in a real window management context. To address this problem, we introduce *Ametista*, a mini-

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

toolkit specifically designed for HCI researchers who want to explore new window management techniques.

RELATED WORK

In this section, we briefly describe the current state of the rendering and windowing systems of Apple Mac OS and Microsoft Windows as well as the X Window system. We also describe several research projects related to the exploration of new window management techniques in a real-use context.

Quartz Compositor

The windowing system of Apple Mac OS X is based on three different libraries: Quartz for 2D graphics, OpenGL for 3D graphics and QuickTime for dynamic media (e.g. animated graphics and video). A fourth component, the Quartz Compositor, is responsible for the composition and display of graphics rendered with these three libraries.

Quartz offers high-quality screen rendering and printing. It is based on the Portable Document Format (PDF) graphics model and features a number of advanced 2D graphics capabilities such as spline-based curves, text rotation, transparency and anti-aliased drawing. The move from the old QuickDraw graphics model to this new one allowed significant visual changes in the user interface of Mac OS. Semi-transparent menus and controls, drop shadows or "fade away" effects that were once limited to a few applications are now available to all through the standard Mac OS graphical user interface toolkit.

The Quartz Compositor is based on the idea that the window system can be considered as a digital image compositor [11]: Quartz, OpenGL and QuickTime graphics are rendered into off-screen buffers that are then used as textures to create the actual on-screen display. As a matter of fact, since Mac OS X v10.2, the Compositor is just another OpenGL application. As such, it can take advantage of hardware-accelerated graphics functions to transform windows in real-time before composing them. Examples of transformations include alpha blending, color fading or geometric transformations, as the *scale* and *genie* effects shown when windows are minimized.

The introduction of Quartz and the Quartz Compositor in the graphics and windowing systems of Mac OS illustrates the potential uses of richer graphical models for supporting new graphical interaction techniques and therefore, new window management techniques. However, the windowing system of Mac OS is tightly coupled with the operating system, which makes it difficult - if not impossible - to access and modify. Although the image compositing approach coupled with hardware-accelerated rendering seems promising, the Quartz Compositor in its current state cannot be used by HCI researchers to explore new window management techniques.

The Task Gallery and Windows Longhorn

Microsoft's Task Gallery [20, 24] uses a redirection mechanism for hosting existing Windows applications in a 3D workspace without changing or recompiling them. By taking advantage of the powerful graphics model of Direct3D, the authors created a 3D window manager that better takes into account the human perception and spatial cognition. This example clearly shows again how a rich graphics model can significantly change the user's interactions with applications and documents.

According to its Web site[1], "the Task Gallery is not a future version of the Windows operating system or user-experience". Yet, recent talks from Microsoft at the Windows Hardware Engineering Conference clearly state that the windowing system of the future versions of Windows will be based on Direct3D and a compositing process [15].

The Task Gallery is a stand-alone application. However, in order to implement their redirection mechanism, the authors had to modify Windows 2000. As they are not allowed to release the patches corresponding to these modifications, the Task Gallery and its redirection mechanism remain out of reach for HCI researchers, like Apple's Quartz Compositor.

The X Window system, Render and RandR

A key feature of the X Window System [22] is that any user-level application can act as a window manager. As a consequence, a large number of window managers have been developed for this system, providing a large range of appearances and behaviors. Yet, all these window managers are based on the original X graphics model and therefore, they differ mostly in minor details such as window decorations or keyboard shortcuts, and not in their operation principle. Most windows remain rectangular and opaque, very little use is made of the advanced features of modern graphics hardware and the interaction techniques and metaphors remain the same.

The mismatch between the original X Window rendering system and modern interactive graphical applications is very well described by K. Packard in [17]: "*The two new open source user interface environments, Gnome and KDE, were hamstrung by the existing X rendering system. KDE accepted the limitations of the environment and made the best of them. Gnome replaced server-side rendering with client-side rendering turning the X protocol into a simple image transport system. The lack of hardware acceleration and the destruction of reasonable remote application performance demonstrated that this direction should be supplanted with something providing a modicum of server-side support.*"

The X Rendering extension (Render) [17] and the Resize and Rotate extension (RandR) [10] were designed to address many of the shortcomings of the original X rendering architecture. These two extensions provide image compositing operators and glyph management and allow applications to resize, rotate, reflect and change the refresh rate of an X screen on the fly. XFree86 4.3.0 partially implements the Render extension, providing anti-aliased text drawing and image composition. Support for the RandR extension has also been partially integrated, providing support for resizing the root window at run-time.

---

[1] http://research.microsoft.com/ui/TaskGallery/

The recent changes in the X Window rendering system coupled with its openness and extensibility makes it more and more usable to explore new graphical interaction techniques. However, basic functions of the Render extension such as affine transformation of images remain to be implemented in existing X servers. Even then, the graphics model of X will still be far simpler than the one of OpenGL, for example. Therefore, OpenGL-based applications will remain graphically richer than any possible X window manager for some time.

VNC-based approaches to new workspace interaction techniques

As we have seen, the graphics and windowing systems of the three most popular platforms make it difficult if not impossible for HCI researchers to take advantage of modern graphics hardware to explore new graphical interaction techniques for window management. In order to overcome these difficulties, a number of researchers are using the VNC remote display system [19] to bring existing desktops and applications into innovative workspaces.

The Three-Dimensional Workspace Manager (3Dwm) [8] includes a VNC viewer implementation that makes it possible to integrate traditional graphical desktops into an immersive 3D environment implemented with OpenGL. In a similar way, Shiozawa et al. use VNC to combine several individual desktops into a perspective layered collaborative workspace [23]. Denoue et al. also used VNC to capture window contents and display them as paper flyers posted on a virtual board [7].

These examples show how VNC can help create innovative workspace interactions without modifying the operating system or the window system. However, in the first two examples, the documents and applications are still displayed and manipulated through the traditional desktop interface, which is simply mapped as a whole inside a new environment. The third example is more interesting regarding window management techniques, although individual windows are captured at regular intervals through a polling mechanism, which, as the authors admit, is not responsive enough to content changes.

GENERAL OVERVIEW OF AMETISTA

Ametista is a mini-toolkit designed to facilitate the exploration of new window management techniques. It supports low-fidelity prototyping, similar to the Director or Tcl/Tk prototypes described in [14] and [2], as well as high-fidelity prototyping using real applications, as in [20].

The current implementation of Ametista supports three types of windows:

- *pseudo-windows* that are randomly-colored rectangles;
- *placeholders* that display a fixed image or a video stream;
- live windows of X Window applications.

Pseudo-windows can be used for low-fidelity prototyping in the early stages of the exploration of a new window management technique. Placeholders can help getting a better idea of the envisioned technique by displaying snapshots or movies of real applications. Finally, live X windows can be used for high-fidelity prototyping and evaluation of the technique. The three kinds of windows can be freely mixed, as shown in Figure 1.

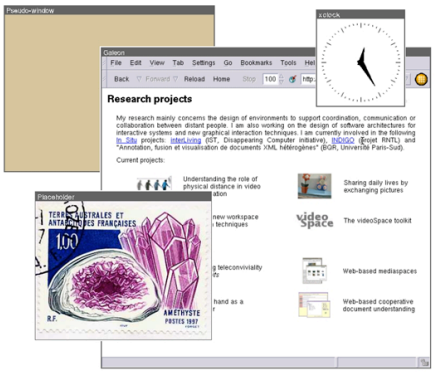


Figure 1: The three window classes of Ametista: a pseudo-window (top-left), a placeholder showing a JPEG image (bottom-left) and two live X Window applications (xclock and the Galeon Web browser).

Ametista uses OpenGL to display windows. As we explained in the previous section, this library offers a rich graphics model well adapted to the exploration of new window management techniques. Alpha blending, for example, makes it easy to create translucent objects and shadows. Scaling, rotation and translation can also be used with a perspective projection to position windows in $2^{1/2}$D or 3D, as illustrated by Figures 2 and 3.

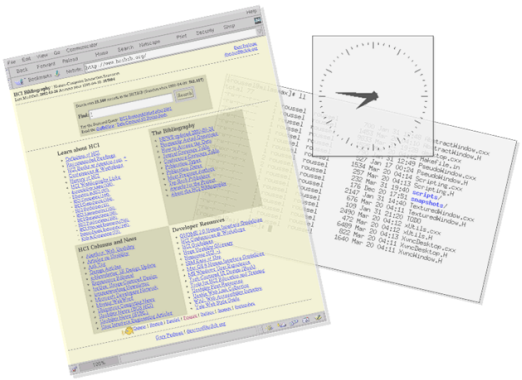


Figure 2: Combining 2D transformations, shadows and transparency.

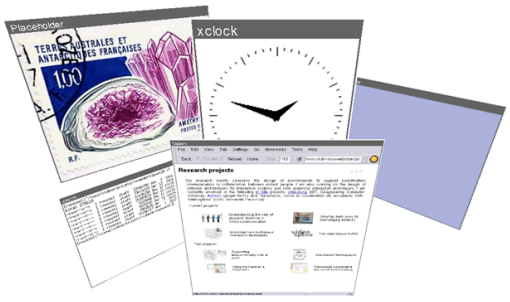


Figure 3: Arranging windows in 3D space.

Ametista makes an extensive use of texture mapping. Textures are used to display fixed images and video streams in placeholders as well as the content of X windows. They also make it possible to transform the window shapes in real-time. Figure 4 shows two examples of such transformations: a peeled back window (Galeon), as described in [2], and two windows cropped to circular shapes (xclock and a placeholder showing a JPEG picture).

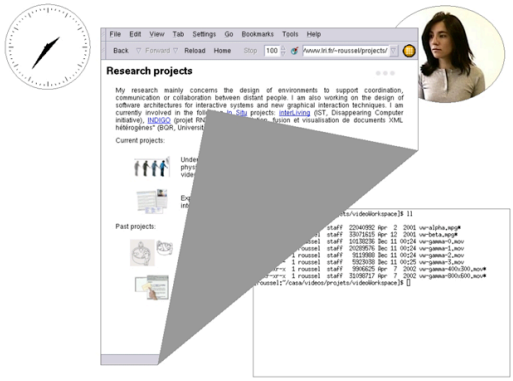


Figure 4: Examples of window shape transformations using texture mapping.

IMPLEMENTATION DETAILS

Ametista is implemented in C++ and uses the videoSpace toolkit [21], OpenGL and VNC. The Ametista software alone consists of about 2500 lines of code. The three window classes described in the previous section (pseudo-window, placeholder and live X window) each correspond to a C++ class that derives from *AbstractWindow*. This class gives access to the content of the window (color or texture) as well as geometry information for mapping screen coordinates to window coordinates.

Each window object has an associated *AbstractWindowRenderer* object. Developers will typically derive this class to redefine methods such as *pointerEvent* or *keyEvent* to manage mouse and keyboard events that occur in the window or *display* to redisplay it when the content has changed. Several renderer classes have been implemented to experiment with transparency, shadows or interactive animations such as the peeling-back effect.

In addition to the implementation of the three window classes and a set of renderers, Ametista also provides the skeleton of a generic window manager that can be customized to implement specific layout and interaction techniques.

X Window application display redirection

We use an approach similar to the redirection mechanism of [24] to make X Window applications available in Ametista. Our approach is based on the X Window version of the VNC remote display system.

VNC consists of two user-level applications: a server that generates the display, and a viewer that draws the display on a screen, receives mouse and keyboard events and forwards them to the server. XVNC, the VNC server implementation for X Window, is a slightly modified but fully functional X server. This server renders applications off-screen, making the desktop image available to VNC viewers (Figure 5), and forwards mouse and keyboard events to the appropriate applications.

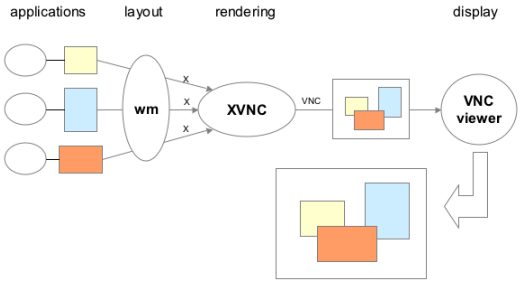


Figure 5: Standard XVNC remote display operation. Note that the window manager (wm) is not part of the VNC system.

The videoSpace toolkit implements the viewer side of VNC as an image source: new desktop images become available from this source whenever display updates are received from the VNC server. This provides Ametista with a real-time stream of images of the X Window desktop[2]. Note that, as opposed to [7], desktop images are pushed by the VNC server to Ametista and not pulled at regular time intervals, which ensures a good response time to application changes.

In order to extract the images of individual applications from desktop images, Ametista also implements the window manager used by the XVNC server. This window

[2] a previous version of Ametista was called VideoWorkspace to reflect the fact that the VNC desktop is seen as a video stream by our compositing process

manager simply tiles the windows next to each other so they do not overlap (Figure 6).

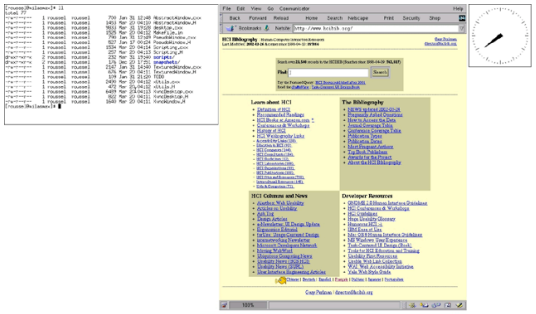


Figure 6: Sample tiled layout of an XVNC desktop managed by Ametista.

Ametista uses the XVNC desktop image as a texture. Whenever it is notified that part of this image has changed, it updates the texture and notifies the corresponding window objects. The *display* method of the renderers associated to these objects uses the size and position communicated by the window manager to set the appropriate texture coordinates. In order to reduce memory usage and achieve better performance, Ametista uses several OpenGL extensions to handle non-power of two textures and to avoid unnecessary memory copies between image data and textures.

Figure 7 summarizes our output redirection mechanism. Note that this approach differs from 3Dwm or the perspective layered workspace from [23], in that Ametista is able to extract images of individual applications and not images of the desktop as a whole. The same effect could be obtained without VNC. For example, we could modify an existing X server, as described in [9], or use a more platform-specific technique.

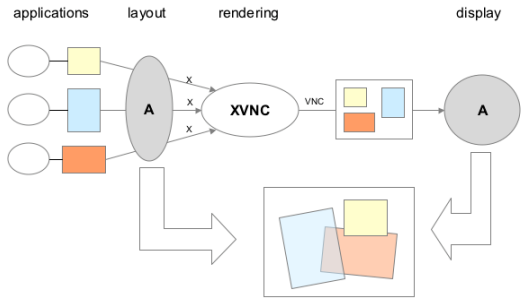


Figure 7: Output redirection of X Window applications using Ametista (VW).

Input handling

Ametista uses OpenGL selection mode and picking to assign the keyboard focus to the window under the mouse. Keyboard and mouse events can be handled locally by the Ametista application, e.g. to implement window management operations such as moving a window. They can also be forwarded to the proper X Window application, in which case the pointer coordinates are transformed into the local window coordinates system.

Window creation and destruction

Ametista can read commands from the standard input to create pseudo-windows and placeholders and to connect to XVNC servers. Each command specifies a rendering class, a window class and some additional parameters such as width, height and title for pseudo-windows or a URL for XVNC servers. As an example, the following commands were executed to set up the windows shown in Figure 1:

```
decorated PseudoWindow 400 300 Pseudo-window
decorated PlaceHolder demo/amethyste.jpg Placeholder
decorated XvncDesktop vnc://127.0.0.1:1
```

When the connection with an XVNC server is first established, all windows existing on this server are automatically added to Ametista's workspace. X applications can then create and destroy windows at will.

A video-enabled application

The videoSpace toolkit provides Ametista with a variety of image sources to be displayed on placeholders. These sources include JPEG and PNG images, QuickTime and MPEG movies but also live video input (e.g. a webcam) as well as networked image sources. As videoSpace image sources are described by URLs, they can be specified at run-time with *PlaceHolder* commands such as the one above.

VideoSpace also provides several image sinks that make it possible to record Ametista's display as a QuickTime or MPEG file or to send it over the network to another application. We already took advantage of this to create short video clips demonstrating Ametista. But most importantly, we anticipate that this feature will be especially interesting for observing users during evaluations and keeping records of these evaluations.

Performance evaluation

We conducted a preliminary evaluation of the performance of Ametista with the images presented in this article. The software ran on a Fujitsu/Siemens PC with a 1.5 GHz Pentium IV and an AGP NVidia GeForce2 MX 400. The operating system was Linux Mandrake 9.0. The screen size was 1280x1024 and the XVNC desktop size was 1280x1024. Ametista achieved full-screen display rates of up to 65 frames per second. Display rates of more than 30 frames per second were also achieved on a 667 MHz Apple PowerBook G4 with an AGP ATI Radeon Mobility.

DISCUSSION

The work presented in this paper relies on the assumption that richer graphics models will allow significant changes in window management techniques in the near future. But what kinds of changes do we expect? In this section, we take several basic features of modern graphics libraries such as OpenGL or Direct3D and explain how we think these features will help us create innovative graphical presentations and interaction techniques for window management.

The third dimension

3D user interfaces are very controversial. On one hand, user studies like [20] show that placing documents and applications in 3D space helps users remember where they are during later retrievals. Yet, other studies like [6] tell us that performance deteriorates as the freedom to locate items in the third dimension increases and that 3D interfaces can be perceived as more cluttered and less efficient than 2D or $2D^{1/2}$. On a less academic perspective, endless discussions about the potential benefits and disadvantages of 3D interfaces (including window managers) are also regularly posted on discussion forums[3].

Most comments in these discussions are related to the frequent navigation problems encountered in 3D interfaces and the need for better input techniques. It is true that a 3D *drag-and-drop* operation on a window might require a lot more concentration and effort than its 2D equivalent. However, specific devices such as isometric joysticks and spaceballs or even better, bi-manual interaction techniques, can help solve these problems.

Many other comments point out that reading, writing and drawing cover a fair amount of our uses of computers and are almost always associated to 2D surfaces. Some people think this makes 3D interfaces inadequate for these tasks. However, when dealing with physical objects, whether 2D or 3D, we perceive them and manipulate them in a 3D world. The same could be true for the digital world. The interesting problem is to find the appropriate interaction techniques and we believe that window management is a good test case for these techniques as it is an unavoidable task.

On a more pragmatic perspective, the third dimension combined with the depth test offers a convenient way to implement multi-layer graphical applications. Each layer can be associated to a particular depth, which can reduce the need for specific data structures. The activation of the depth test allows to render objects in arbitrary order, pixels being updated only if the current object is closer than the one already displayed (if any). Ametista already uses this approach to display overlapped windows, assigning a different depth to each window.

Geometric transformations

Moving windows (translating them) has always been possible since the adoption of the overlapping model. Scale transformations have almost never been possible. Note that the resizing of a window is usually not a scale transformation since it changes the layout of the window instead of just making the content bigger or smaller. The closest things to scale transformations of windows are the icons used in several systems that show a reduced version of the original application display. Rotations of windows have never been possible until the RandR extension of the X Window system that allows to change the orientation of the whole display.

Scale transformations have been used to create zoomable interfaces for a while [18]. While scaling the whole workspace might not be a good idea, we believe that scaling individual windows will be much more interesting. Translations of objects combined with a perspective view allow to *move away* some objects, making them smaller, and *bring closer* some others. We are currently using Ametista to explore this kind of interactions with windows (Figure 8).

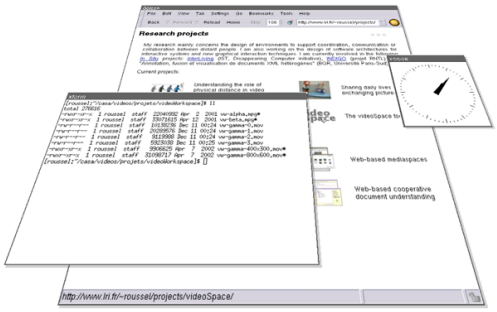


Figure 8: Example of perspective view.

We anticipate that rotations might play an important role in the future. In situations like Figure 8, viewpoint rotations allow to explore the three dimensions in a continuous way. As illustrated by Figure 2, rotations might also be used to better differentiate windows. In that case, similar orientations could be used to indicate that two windows are related in some way (e.g. they belong to the same application or they refer to the same document). Rotations of individual objects also make it possible to create interfaces for horizontal displays, which are particularly interesting for single-display groupware situations [25].

Alpha blending

Alpha blending allows to easily create translucent objects. As illustrated by Figure 2, we have started experimenting with the use of translucency for window contents and decorations. The least we can say after these initial tests is that it is not clear what windows should be made transparent, why and for how long. Obviously, the interesting property of a translucent object is that one can see through it. Thus, translucency should be valuable when one wants to see something behind the current object of interest. This suggests that translucency might be better thought of as a time-limited interaction technique rather than a timeless property of an object. This, in turn, might explain why ten years after the publication of the first paper describing them [4], toolglasses and magic lenses are still the best examples of use of alpha blending in graphical interfaces.

[3] check http://www.useit.com/alertbox/981115.html, http://slashdot.org/article.pl?sid=99/11/03/0917216 or http://nooface.com/search.pl?topic=visualui for some examples of these discussions

Alpha blending also poses a number of pragmatic problems. Primitives drawn using it need to be drawn after all opaque primitives are drawn. Moreover, unless the translucent objects are sorted in back-to-front order, depth buffer updates must be disabled, although depth buffer compares should remain enabled. Maintaining this back-to-front sorted list can be quite expensive if many geometric transformations are applied on the objects.

Texture mapping, lighting and image processing

Figure 4 illustrates how texture mapping can be used to transform window shapes. More complex transformations could be easily implemented in Ametista. As an example, one could create a window renderer that would apply a fisheye deformation on the window's content. One could also implement a renderer that would display only a part of the content that would have been selected interactively (the circular crop shown in Figure 4 is computed automatically).

The current implementation of Ametista does not make any use of lighting. Yet, material lighting and shading could be used, for example, to highlight the window having the keyboard focus. Similarly, image processing techniques could be used to render some windows out of focus to get a sense of depth of field. Full screen antialiasing or motion blur could also be implemented through these techniques.

The recent introduction of programmable shaders in Direct3D and OpenGL has made visual quality of interactive computer graphics take a quantum leap towards realism. We anticipate that these shaders will help us create new rendering transformations and filters for Ametista in the future.

CONCLUSIONS AND FUTURE WORK

We have presented Ametista, a mini-toolkit for exploring new window management techniques. We have described how this toolkit supports both the low-fidelity prototyping of these techniques using pseudo-windows and placeholders as well as a high-fidelity approach based on X Window application redirection. Preliminary results are encouraging: we have been able to use Ametista to experiment with several rendering styles and interaction techniques with excellent performance.

Future work on Ametista includes a better tiling algorithm for the XVNC window manager and more scripting capabilities. We plan to use the toolkit to implement and evaluate some of the layout algorithms, interaction techniques and metaphors contributed by the HCI community. Of course, we also plan to use it to explore some of the directions we mentioned in the previous section.

AVAILABILITY

Ametista and videoSpace are available in source code from

http://www.lri.fr/~roussel/software/

Several short videos of Ametista are also available from

http://www.lri.fr/~roussel/projects/ametista/

ACKNOWLEDGEMENTS

The author thanks Wendy Mackay, Michel Beaudouin-Lafon, Olivier Beaudoux, Renaud Blanch and Stéphane Conversy for providing helpful comments on an earlier version of this document.

REFERENCES

1. G. Badros, J. Nichols, and A. Borning. *Scwm - an intelligent constraint-enabled window manager*. In proceedings of AAAI Spring Symposium on Smart Graphics. IEEE Computer Society Press, March 2000.
2. M. Beaudouin-Lafon. *Novel interaction techniques for overlapping windows*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2001, Orlando (USA), pages 153-154. ACM Press, November 2001.
3. M. Beaudouin-Lafon and H.M. Lassen. *The architecture and implementation of cpn2000, a post-wimp graphical application*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2000, San Diego (USA), pages 181-190. ACM Press, November 2000.
4. E. Bier, M. Stone, K. Pier, W. Buxton, and T. De Rose. *Toolglass and magic lenses: the see-through interface*. In Proceedings of ACM SIGGRAPH 1993, pages 73-80. ACM Press, 1993.
5. S.A. Bly and J.K. Rosenberg. *A comparison of tiled and overlapping windows*. In Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems, pages 101-106. ACM Press, 1986.
6. A. Cockburn and B. McKenzie. *Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments*. *CHI letters*, 4(1):203-210, April 2002. Proceedings of ACM CHI 2002 Conference on Human Factors in Computing Systems, Minneapolis.
7. L. Denoue, L. Nelson, and E. Churchill. *Attractive windows: Dynamic windows for digital bulletin boards*. Conference companion, Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems, to be published (2 pages).
8. N. Elmqvist. *3Dwm: Three-Dimensional User Interfaces Using Fast Constructive Solid Geometry*. Master's thesis, Chalmers University of Technology, Göteborg, 2001.
9. S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon. *Windows on the world: 2d windows for 3d augmented reality*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST '93, Atlanta (USA), pages 145-155. ACM Press, November 1993.
10. J. Gettys and K. Packard. *The X Resize and Rotate Extension - RandR*. In proceedings of USENIX Annual Technical Conference, FREENIX Track, pages 235-243. USENIX association, 2001.
11. P. Graffagnino. *Apple OpenGL and Quartz Extreme*. Presentation at SIGGRAPH 2002, OpenGL BOF.

12. J. Johnson, T.L. Roberts, W. Verplank, D.C. Smith, C. Irby, M. Beard, and K. Mackey. *The Xerox Star: a retrospective*. IEEE Computer, 22(9):11-29, September 1989.

13. E. Kandogan and B. Shneiderman. *Elastic Windows: evaluation of multi-window operations*. In Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems, Atlanta, pages 250-257. ACM Press, March 1997.

14. R. Mander, G. Salomon, and Y.-Y. Wong. *A pile metaphor for supporting casual organization of information*. In Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems, pages 627-634. ACM Press, 1992.

15. C. McCartney. *Windows Desktop Composition*. Presentation at WinHEC 2002, Windows Graphics Architecture session.

16. B.A. Myers. *A taxonomy of window manager user interfaces*. IEEE Computer Graphics and Applications, 8(5):65-84, sept/oct 1988.

17. K. Packard. *Design and Implementation of the X Rendering Extension*. In Proceedings of USENIX Annual Technical Conference, FREENIX Track, pages 213-224. USENIX association, 2001.

18. K. Perlin and D. Fox. *Pad: An alternative approach to the computer interface*. In Proc. of ACM SIGGRAPH 1993, pages 57-64. ACM Press, 1993.

19. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. *Virtual Network Computing*. IEEE Internet Computing, 2(1):33-38, Jan-Feb 1998.

20. 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*. In Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems, pages 494-501. ACM Press, April 2000.

21. N. Roussel. *Exploring new uses of video with videoSpace*. In R. Little and L Nigay, editors, Proceedings of EHCI'01, the 8th IFIP International Conference on Engineering for Human-Computer Interaction, volume 2254 of Lecture Notes in Computer Science, pages 73-90. Springer, 2001.

22. R.W. Scheifler and J. Gettys. *The X Window system*. ACM Transactions on Graphics, 5(2):79-109, 1986.

23. H. Shiozawa, K. Okada, and Y. Matsushita. *Perspective layered visualization of collaborative workspaces*. In Proceedings of the international ACM SIGGROUP conference on supporting group work, pages 71-80. ACM Press, November 1999.

24. M. van Dantzich, G. Robertson, and V. Ghorokhovsky. *Application Redirection: Hosting Windows Applications in 3D*. In Proceedings of NPIV99, the workshop on New Paradigms on Information Visualization and Manipulation, pages 87-91. ACM Press, 1999.

25. F. Vernier, N. Lesh, and C. Shen. *Visualization techniques for circular tabletop interfaces*. In Proceedings of AVI'2002, Trento, Italy, pages 257-263, May 2002.

Metisse is not a 3D Desktop!

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

ABSTRACT
Twenty years after the general adoption of overlapping windows and the desktop metaphor, modern window systems differ mainly in minor details such as window decorations or mouse and keyboard bindings. 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. We believe that one reason for this is that most of the proposed techniques have been designed using a low fidelity approach and were never made properly available. 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. We describe the architecture of the system, some of its implementation details and present several examples that illustrate its potential.

ACM Classification: H.5.2 [User Interfaces]: Windowing systems. D.4.9 [Systems Programs and Utilities]: Window managers.

General terms: Design, Human Factors.

Keywords: Window system, Window management.

INTRODUCTION
Overlapping windows that users can freely move and resize have been described more than thirty years ago and have been available to the general public for more than twenty years [19]. Over time, various interaction techniques have been proposed to control the placement, size and appearance of application windows. Yet, from a user perspective, the most popular window systems differ mainly in minor details such as window decorations or mouse and keyboard bindings, and not in their fundamental operation principles. As Myers already put it in 1988, "there is not a great deal of difference among different window managers" [18].

The growing range of activities supported by interactive computer applications makes it more and more difficult to remember these activities and to organize them. At the same time, recent advances in computer graphics and display technologies combined with decreasing costs are changing the nature of the problem. High performance graphics cards, high definition displays, big screens and multiple monitor systems are becoming common place. From the time when window systems were too demanding and had to be carefully tuned for performance, we have now moved to a situation where a lot of software and hardware resources are available. The question is: how should these resources be used?

Little research has been performed on understanding people's space management practices [13]. While a number of innovative window management techniques have been proposed by HCI researchers over the last few years [29], very few of these techniques have been formally evaluated and even fewer have made their way into current window systems. We believe that these two points are strongly related to the fact that most of the techniques proposed by the HCI community were designed using a low fidelity approach and were never made properly available in a real window system.

Building a whole new window system is a hard task, one that few HCI researchers are willing to do. At the same time, existing systems are either closed boxes, inaccessible to developers, too limited for the envisioned interaction techniques or too complex to program. How would you implement a zoomable window manager? One that would strengthen the paper and desktop metaphor? One that could be used on an interactive table? One that would support bi-manual interaction?

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. Metisse uses an image compositing approach that makes it possible to apply a number of visual effects and geometrical transformations on windows. But Metisse is not a 3D desktop. It is a "meta window-manager", an enabling tool for window management research.

The paper is organized as follows. After introducing some related work, we describe Metisse by providing an overview of its design and architecture as well as some implementation details. We then present several examples that illustrate its potential for exploring new window management techniques. Finally, we conclude with a discussion and some directions for future research.

UIST'05, October 23-27, 2005, Seattle, Washington, USA.

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

RELATED WORK

In this section, we briefly describe the current state of the three most popular window systems as well as several research projects related to the exploration of new window management techniques.

Apple Mac OS X, Microsoft Windows and X Window

The Apple Mac OS X graphics system is based on three different libraries : Quartz for 2D graphics (a rendering engine based on the PDF drawing model), OpenGL for 3D graphics and QuickTime for animated graphics and video. Windowing services are available through a software called *Quartz Compositor* [1]. This software handles the compositing of all visible content on the user's desktop: Quartz, OpenGL and QuickTime graphics are rendered into off-screen buffers that the compositor uses as textures to create the actual on-screen display.

Among other features, Quartz Compositor supports window transparency, drop shadows and animated window transformations, which are used to create various effects such as the *scale* and *genie* effects used for window iconification, the *fast user switching* animation, the three *Exposé* modes and other *Dashboard* effects. From a developer perspective, however, Quartz Compositor is a closed box. Most of its functionalities are available through a private, undocumented and probably unstable API that only a few highly-motivated developers are willing to use [1]. Gadget applications using this private API are interesting because they show that the compositor is much more powerful than it seems and that services, such as *Exposé*, are in fact the result of a careful selection of its features. At the same time, this is very frustrating since this compositing policy and the associated design space remain out of reach for the HCI researcher.

The window system of Microsoft Windows is tightly coupled with the operating system, which makes it difficult to access and modify. Several applications such as SphereXP[2] replace the traditional desktop by a 3D space in which arbitrary objects can be painted with 2D images from application windows. However, the implementation details of these systems are not available. The next version of Microsoft Windows will most probably include a composite desktop based on DirectX 9 [3]. Details of what will be available to users and developers remain uncertain. However, one can reasonably imagine that the compositing policy will probably be out of reach for the average developer and HCI researchers.

A key feature of the X Window System [25] (or X) is that any application can act as a window manager. As a consequence, a large number of window managers have been developed for this system, providing a range of appearances and behaviors. Recent X extensions make it now possible for these window managers to use a compositing approach [10]: *Composite*, that allows windows to be rendered off-screen and accessed as images; *Damage*, that allows an application to be notified when window regions are updated; and *Event Interception*, that allows keyboard and mouse events to be pre-processed before being sent to their usual targets. Another extension, *Xfixes*, provides the data types and functions required by these extensions.

Experimental X window managers are slowly taking advantage of these extensions to provide visually attractive effects similar to the ones proposed by Mac OS X. However, the numerous extensions required make it hard for developers unfamiliar with the X architecture to implement their own compositing window manager. Moreover, implementing a fully functional and standard-compliant X window manager requires much more than simple window image compositing and event pre-processing.

Window management research

Many of the window management solutions proposed by the HCI research community have been designed using a low-fidelity approach and have never been implemented as part of a real window system. *Elastic windows* [16], for example, were only implemented within custom applications. *Peeled back windows* have been demonstrated within specific Tcl/Tk and Java prototypes [2, 9]. Window shrinking operations and dynamic space management techniques have also been demonstrated within specific Java prototypes [14, 4].

A notable exception to the low-fidelity approach is the Rooms system [11]. Designed in 1985 by Card and Henderson based on an analysis of window usage [6], this system was originally implemented in Interlisp-D on Xerox D-machines. It was ported to C and the X Window environment in 1989, and to Microsoft Windows in the early 1990s. Rooms (or *virtual desktops*) have since gained a fairly widespread acceptance: they are now supported by most X window managers and are also available on Windows XP and Mac OS X through additional utilities. Rooms are probably the best example of the potential impact of window management research.

A second notable exception is Microsoft's Task Gallery [23], a system that uses input and output redirection mechanisms for hosting existing Windows applications in a 3D workspace. The redirection mechanisms require several modifications of the standard window manager of Windows 2000. They provide off-screen rendering and event pre-processing facilities similar to those becoming available in X [30]. However, in this case, since the Windows 2000 modifications have never been publicly released, very few people outside Microsoft have been able to experiment with this system.

Several recent projects have tried to move from low-fidelity prototypes to real functional systems. Scalable Fabric [22], mudibo [15] and WinCuts [28], for example, are implemented as "real" applications supplementing the legacy window manager of Windows XP. However the fact that these systems are developed outside the window system makes them unnecessarily complex, potentially inefficient and harder to combine with other window or task management techniques. As an example, since they can't be notified of window content updates, the three mentioned systems resort to periodically calling a slow *PrintWindow* function to get window images, which is both inefficient and unsuitable for interactive manipulation of the window content.

[1] `http://cocoadev.com/index.pl?WildWindows`
[2] `http://www.hamar.sk/sphere/`

Experimental desktop environments

Ametista [24] is a mini-toolkit designed to facilitate the exploration of new window management techniques. It supports the creation of new OpenGL-based desktop environments using both a low-fidelity approach, using placeholders, as well as a high-fidelity approach based on X application redirection through a custom implementation of the VNC [21] protocol. Several 3D environments such as Sun's Looking Glass[3]and Croquet [26] are also using the X extensions we already mentioned to host existing applications.

The main problem of these new environments is that although they provide the fundamental mechanisms for implementing compositing window managers, they implement only parts of the standard X protocols related to window management (e.g. ICCCM, EWMH) that define the interactions between window managers, applications, and the various utilities that constitute traditional desktop environments such as GNOME or KDE. As a consequence, these environments are hardly usable on a daily basis, since they do not support common applications such as mail readers, Web browsers, media players or productivity tools.

METISSE

Metisse is an X-based window system designed with two goals in mind. First, it should make it easy for HCI researchers to design and implement innovative window management techniques. Second, it should conform to existing standards and be robust and efficient enough to be used on a daily basis, making it a suitable platform for the evaluation of the proposed techniques. Metisse is not focused on a particular kind of interaction (e.g. 3D) and should not be seen as a new desktop proposal. It is rather a tool for creating new types of desktop environments.

The design of Metisse follows the compositing approach and makes a clear distinction between the rendering and the interactive compositing process. The *Metisse server* is a modified X server that can render application windows off-screen. The default compositor is a combination of a slightly modified version of a standard X window manager, *FVWM*, with an interactive viewer called *FvwmCompositor*. As we will see, the use of FVWM provides a lot more flexibility and reliability than custom-made window managers such as those used by Ametista or Looking Glass. We will also show that other compositors can be used in conjunction with the Metisse server.

Metisse is implemented in C and C++ and runs on the Linux and Mac OS X platforms. Figure 1 shows the communication links between the various software components. The Metisse server sends window-related information, including window images, to FvwmCompositor. FvwmCompositor displays these images with OpenGL, using arbitrarily transformed textured polygons, and forwards input device events to the server. FVWM can solely handle basic window operations such as move, resize or iconify, issuing the appropriate commands to the X server. It can also delegate these operations to FvwmCompositor. New window operations can be implemented either as FVWM functions, using a scripting language, or in FvwmCompositor.

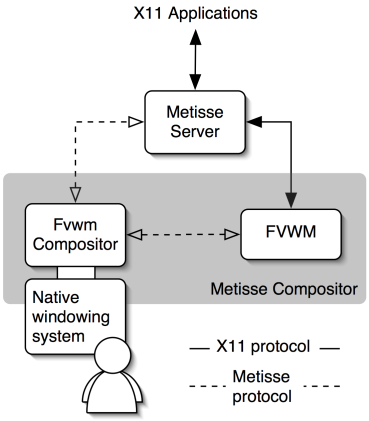


Figure 1: General overview of Metisse.

The following subsections will provide more implementation details about the Metisse server, our modified FVWM and FvwmCompositor.

Metisse server

The Metisse server is a fully functional X Window server derived from Xserver[4], software used by the X community for exploratory developments. It uses Xserver's rootless extension to provide off-screen rendering of application windows: each top-level window is rendered in a separate pixmap – an image stored in a single contiguous memory buffer – that is dynamically allocated when the window is created and reallocated when it is resized. The server stores along with each window image the coordinates of its upper-left corner in the compositor's display. These coordinates are the ones reported to applications that issue geometry requests.

Each time an application updates a window content, the server sends an update notification to the compositor. The corresponding region of the pixmap can be transmitted to the compositor using a custom protocol similar to VNC. It can also be copied in a shared memory space if the two processes are running on the same machine. These off-screen rendering and update notification mechanisms are quite similar to the *Composite* and *Damage* X extensions. In fact, the main reason why we didn't use these extensions is that they were still in early development stages and heavily discussed when we started implementing Metisse.

The server sends various other notifications to the compositor to indicate the creation, destruction, mapping or unmapping of a window as well as geometry modifications, changes in the stacking order and cursor changes. It provides the compositor with the actual bounds of shaped (i.e. non rectangular) windows. It also indicates a possibly related window for all transient and override redirect windows (e.g. pop-up menus). All these notifications make it easy for the compositor to maintain a list of the windows to be displayed and to

[3] http://wwws.sun.com/software/looking_glass/
[4] http://freedesktop.org/Software/Xserver

apply to pop-up menus the transformation used for the corresponding application window.

Window visibility is usually an important concern for X server implementations, since a window can be partially occluded by another one, or be partially off-screen. Traditional servers generate *Expose* events to notify applications of visibility changes and clip drawing commands to the visible regions. Since the Metisse server renders windows in separate pixmaps, partial occlusion never happens. Moreover, since the actual layout of windows is defined by the compositor, the notion of being partially off-screen doesn't make any sense in the server. As a consequence, the Metisse server never generates *Expose* events.

Traditional X servers receiving a mouse event use the pointer location and their knowledge of the screen layout to decide which window should receive the event. Again, in the case of Metisse, since the actual layout is defined by the compositor, the server cannot perform this computation. As a consequence, the mouse events transmitted by the compositor must explicitly specify the target window. When the screen layout changes, X servers usually look for the window under the pointer and, if it has changed, send *Leave* and *Enter* events to the appropriate windows. In the case of Metisse, this process is left to the compositor.

Metisse compositor: FVWM and FvwmCompositor

FVWM[5] is an X window manager created in 1993 and still actively developed. Originally designed to minimize memory consumption, it provides a number of interesting features such as GNOME and KDE compatibility, customizable window decorations, virtual desktops, keyboard accelerators, dynamic menus, mouse gesture recognition, as well as various focus policies. All these features can be dynamically configured at run-time using various scripting languages.

Scripted functions are a powerful and simple way of extending the window manager. As an example, one can easily define a new iconification function that raises the window, takes a screenshot of it with an external program, defines this image as the window icon and then calls the standard iconification command. Commands can be executed conditionally, depending on the nature and state of a window, and can be applied to a specific set of windows. Commands and scripted functions can be easily bound to a particular mouse or keyboard event on the desktop, a window or a decoration element. They can also be bound to higher-level events such as window creation or focus changes.

FVWM can also be extended by implementing *modules*, external applications spawned by the window manager with a two-way communication link. FvwmCompositor is an FVWM module implemented with the Núcleo[6] toolkit, which provides a simple OpenGL scenegraph and a basic asynchronous scheduler for multiplexing event sources and reactive objects. It uses the window images as textures that can be mapped on arbitrary polygons, these polygons being themselves arbitrarily transformed (Figure 2).

[5] http://www.fvwm.org/
[6] http://insitu.lri.fr/~roussel/projects/nucleo/

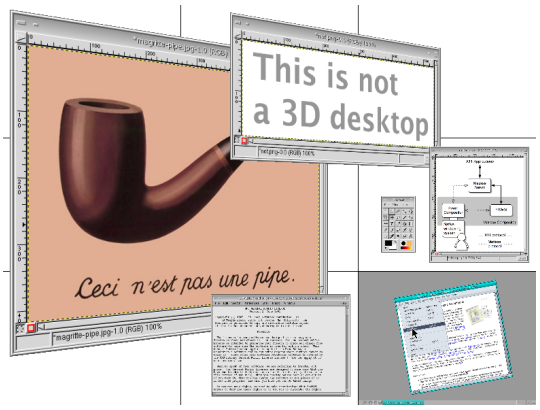


Figure 2: Basic composition showing rotated and scaled windows.

Implementing the Metisse compositor as an extension of FVWM has several advantages over developing one from scratch. First, almost nothing needs to be done to replicate the standard window operations of existing window systems: FvwmCompositor simply needs to display window images at the positions given by the Metisse server, and to forward input device events to it. Second, since FVWM reparents application windows in new ones containing the decorations, these decorations are automatically made available in the compositor through the server. This has proved to be much more convenient than writing OpenGL code to display and interact with title bars and borders, buttons and pull-down menus.

FvwmCompositor displays its composition in an OpenGL window of the native window system (a GLX window on Linux and a Carbon/AGL window on Mac OS X). Although not mandatory, this window is usually set to be full-screen, so that FvwmCompositor visually replaces the native window system. The current implementation uses a perspective projection. The third dimension (i.e. Z axis) of OpenGL is used to enforce the stacking order of the windows defined in the server by FVWM. In order to avoid intersections between windows that would not be on parallel planes, large Z distances are used between windows, especially for the bottom and top ones. Consequently, all windows are rescaled to keep their original size despite their distance to the viewer and the perspective projection.

Keyboard events are simply forwarded to the Metisse server, the keyboard focus policy being handled by FVWM. When receiving a mouse event, FvwmCompositor uses OpenGL's selection mode and picking to find the window under the pointer. It then uses the transformation matrix associated to that window and its position on the server's virtual screen to transform the mouse coordinates into the server's coordinate system. The event is then forwarded to the server with these adjusted coordinates and additional information specifying the target window.

In some situations, FvwmCompositor needs to use the transformation matrix of a particular window even if the mouse

pointer is not over it. This happens in some rare cases under heavy load when the user is interactively resizing the window in a movement so fast that the pointer leaves the window (i.e. the resize operation lags a few steps behind the movement of the pointer). To avoid this particular situation, FvwmCompositor scales up all the polygons when drawing windows in selection mode.

EXAMPLES

In the previous section, we have described the architecture of Metisse and explained how this design provides a window system that is both fully functional and highly tailorable. In this section, we present several examples that illustrate how Metisse facilitates the implementation of innovative window management techniques.

Basic operations

The following code sample shows how FVWM can be configured to scale windows by clicking on one of the buttons of their title bar or pressing some keys:

```
Mouse 3 4 A SendToModule FvwmCompositor Scale 0.7

Key minus W C SendToModule FvwmCompositor Scale 0.9
Key plus W C SendToModule FvwmCompositor Scale 1.11
```

The first line requests FVWM to send the string “Scale 0.7” to FvwmCompositor when the user does a right mouse click (third button) on the minimize icon of the title bar (fourth icon), no matter the active keyboard modifiers (A is for “any modifier”). In addition to the specified string, FVWM will send the X id of the window that received the click. A simple parser implemented in FvwmCompositor will decode the message and perform a 30% reduction of the representation of the specified window. Similarly, the two other lines request FVWM to send a scale command to FvwmCompositor when the user presses Ctrl+ or Ctrl- while the mouse pointer is on the window.

Other commands implemented in FvwmCompositor allow users to rotate a window around different axes, to scale it non-uniformly and to set, lower and raise its opacity and brightness levels. Metisse provides a default configuration file for FVWM with menus and various bindings (mouse, keyboard or high-level events) for all these operations. These bindings make it possible, for example, to lower the brightness of all windows except those of the application having the keyboard focus. One can also specify that all windows of a certain type should be semi-transparent (e.g. those of an instant messaging application) and become fully opaque when the mouse pointer comes over them.

FvwmCompositor commands can also be combined with traditional window operations in interesting ways. As a first example, one can easily replace the usual maximizing operation by a function that zooms in the window so that it takes the whole screen space instead of resizing it. Another interesting combination is the *ZoomOutAndMaximizeHeight* function illustrated by Figure 3. This function zooms out a window uniformly and then resizes its height so that it takes the whole screen. It is available in Metisse as a toggle switch (i.e. calling the function a second time returns the window back to its previous state).

The ZoomOutAndMaximizeHeight function is particularly interesting when working on a large text document. When activated, it makes it possible to see a larger part of the document which in turn makes it easier to navigate. Calling the function a second time restores the original scale and size of the window, providing a more detailed view of the selected part of the document. A notable fact about this function is that it has been designed and implemented in a few minutes on a laptop during a subway trip between Paris and Orsay (about 35 minutes). The final implementation is only a few lines long to be added to the configuration file of Metisse.

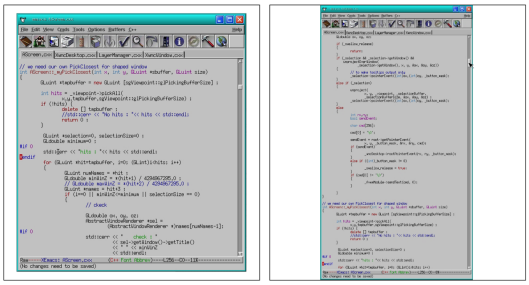


Figure 3: ZoomOutAndMaximizeHeight function applied on a text editor showing a large document. The left image shows the window before calling the function, the right one shows the resulting transformation.

Interactive window manipulation

Interactive window manipulations such as the folding operation described in [2] (Figure 4) cannot be implemented with FVWM scripts. This kind of complex operations need to be handled directly by FvwmCompositor. In order to facilitate this, we have created a new FVWM command called *MetisseDelegate*.

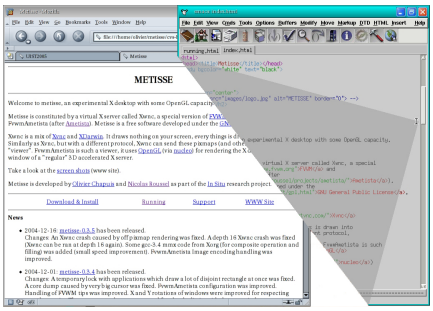


Figure 4: XEmacs window being peeled back.

MetisseDelegate abstracts the concept of interactive manipulation from the FVWM point of view. It takes a window operation name and a cursor name as arguments. When executed, it grabs the mouse and the keyboard (i.e. forbids other applications to use them), changes the cursor for the one specified, and checks that the specified window is still

valid. FVWM then sends to FvwmCompositor the operation name along with the cursor position and the target window, and enters a simple event loop, waiting for the operation to complete. Upon completion, FVWM releases the mouse and keyboard and reenters its main loop.

Interactive move, rotation and scaling are implemented in the same way, as FvwmCompositor operations called in response to an FVWM message sent by MetisseDelegate. Here's how the folding operation might be configured in FVWM:

```
# Immediately (I) raise the window and start the
# fold operation (Fold) if the mouse is dragged (M)
# or hold (H)
AddToFunc FoldWindow
+ I Raise
+ M MetisseDelegate Fold FOLD_CURSOR
+ H MetisseDelegate Fold FOLD_CURSOR

# Bind the folding function to a right mouse button
# click (3) on the window border (F A)
Mouse 3 F A  FoldWindow
```

The interactive scale operation is in a certain way similar to the usual resize operation and can be bound to the manipulation of the borders of the windows with a given keyboard modifier. Rotation around the Y axis can be bound to a mouse drag on the left or right border of the window with a keyboard modifier. The top and bottom borders can be used for rotations around the X axis. The corners of the window might be used for rotations around the Z axis.

We believe that window scaling might offer some interesting new ways of managing overlapping windows. As opposed to the traditional resize operation, scaling a window reduces overlapping while preserving the layout of window contents. This has proved to be useful, for example, for checking the layout of a Web page in one or more browsers while editing it in a text editor. Temporarily scaling down two applications can also make it easier to quickly perform a series of interactions between them, such as drag-and-drop or copy/paste operations. Note that when using a perspective projection, rotations around the X and Y axis produce a non-uniform scaling effect that can also be used to reduce overlapping.

Unlike low-fidelity environments usually used to implement innovative window management techniques, Metisse allows all these techniques to be used for real on a daily basis. This can help adjusting the details of a particular technique. The folding operation, for example, became much more interesting after we decided to make the back side of the peeled-back window translucent (Figure 4). Daily use also helped realize that the ability to put back windows into a "normal" state after some transformation was very important. As a consequence, the default Metisse configuration allows users to cancel the transformations applied to a window by right-clicking on its title bar, a simple animation being used to ease the transition. We are also adding a history mechanism with an undo/redo mechanism that should make it easier to understand manipulation errors and to capture interaction sequences to create new commands.

Animations and temporary transformations

Animations have long been used in window managers to provide feedback of ongoing operations. Specifying simple animations in Metisse is quite easy. It doesn't require much programming skills and could probably be done by experienced users. The following code sample shows how to create an animated iconification function. It uses a `for` loop to send multiple *Scale* commands to FvwmCompositor to produce the animation effect. Note that since this animation is implemented as an FVWM script, it is parsed and executed at run-time and doesn't require any modification of FvwmCompositor.

```
AddToFunc myIconify
+ I PipeRead 'for ((i=0; i<20; i++)) do \
  echo "SendToModule FvwmCompositor Scale 0.9"; \
  done
+ I State 1 True

AddToFunc myDeIconify
+ I PipeRead 'for ((i=0; i<20; i++)) do
  echo "SendToModule FvwmCompositor Scale 1.11"; \
  done
+ I State 1 False

AddToFunc myToggleIconify
# deiconify if iconified
+ I ThisWindow (State 1) myDeIconify
# iconify if not iconified
+ I TestRc (NoMatch)  myIconify
```

Asynchronous animations can also be implemented as scripts by delaying individual command execution with a *Schedule* command (e.g. `Schedule 50ms SendToModule FvwmCompositor Scale 0.9`). However, traditional animation effects like slow-in, slow-out, anticipation or follow through [17, 7] should rather be implemented in FvwmCompositor. Although the Núcleo toolkit cannot guarantee a particular framerate, its asynchronous scheduler makes it possible to follow an approach similar to the one described in [12] for supporting flexible and reusable animation.

The functions for moving, rotating and scaling windows have been implemented in two forms. The first one corresponds to the usual operation mode: the user presses a mouse button and moves the mouse to define the transformation. When the button is released, the window stays where it is, using the last transformation. The second form is a temporary elastic one: when the mouse button is released, the window returns to its original position with an animation. This second form provides interesting alternatives to the folding operation. While experimenting with translucency effects, we also found out that temporary modifications of the opacity level also offers new interesting possibilities. As an example, when moving a window, making that window or the other ones translucent helps finding the right place to put it.

Position-dependent window transformations

Until now, we have presented techniques that put the user in total control of every detail of the transformations applied on windows. However, combining two or more elementary transformations, such as a move and a rotation, can be quite tedious. A simple and powerful way of solving this problem is to define the transformation to be applied on a window as

dependent of its position. This way, the user will indirectly apply these transformations by simply moving the window.

As a first example, we used this approach to scale down windows as they approach the right border of the screen, so that they remain fully visible instead of becoming partly off-screen (Figure 5). A minimum size is imposed so that at some point, trying to move the window further has no effect anymore. A special FvwmCompositor command restores the original position and size of a window before it was moved (and scaled) to the side. As opposed to the usual iconification operation, this new operation provides a simple and continuous way of moving a window from a focus region to the periphery. Although different in spirit, it is in some ways similar to the window manipulation techniques provided by Scalable Fabric [22][7].

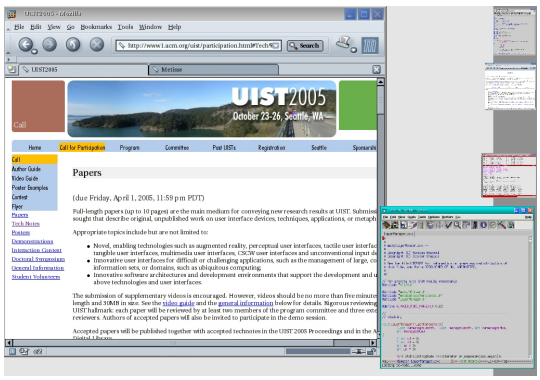


Figure 5: Windows on the right side of the image have been scaled down as they were pushed towards the border of the screen.

Daily use of this move-and-scale operation also gave us the idea of implementing a second version of it, a temporary one following the approach described in the previous subsection. This version allows users to grab a window with the mouse, move it to the side of the screen (and thus reduce its size) and then simply release the mouse button to restore the window's original size and position. This new operation proved to be quite useful to see what's behind a window while keeping its content visible.

Our second example of position-dependent transformation has been designed for tabletop interactive displays [8]. In this example, we split the screen into two equal parts A (the bottom part) and B (the top part). When a window is totally contained in A no transformation is applied to it. When it is totally contained in part B, it is rotated around the Z axis by 180 degrees. When a window is between the A and B parts a rotation between 0 and 180 degree is applied to it depending on the distance to the splitting line. The rotation is applied clockwise if the center of the window is on the left part of the screen and counterclockwise if on the right. This way, if a window is moved from A to B, it is progressively rotated upside down (Figure 6).

One nice feature of FvwmCompositor is that it can duplicate windows. Users can interact with a duplicated window exactly as if it were the original one (see [27] for more details). This feature can be combined with the tabletop interactions we just described: the top-left window of Figure 6 is a zoomed duplicate of the one in the middle of the lower part. Window duplication is also interesting in multiple-monitors configurations. Although the current implementation of Metisse does not support multiple simultaneous input, this example has already proved to be useful in situations where one user needs to show something to other people.

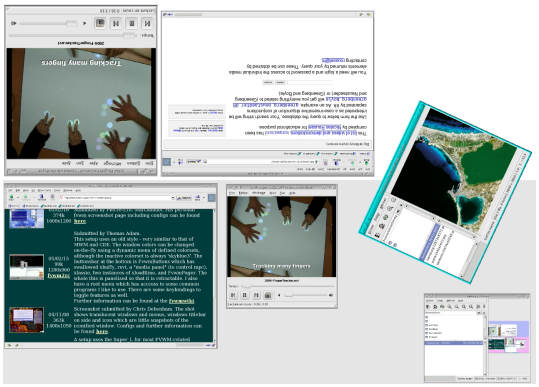


Figure 6: Tabletop interface featuring automatic window orientation and on-demand window duplication.

Interactive desktop manipulation

Global operations that transform all the windows can also be implemented in Metisse. As an example, we have implemented a zoomable desktop nine times bigger than the physical screen (nine virtual screens arranged in a 3x3 matrix). This desktop supports standard panning techniques to move from one virtual screen to adjacent ones by simply moving the mouse towards the corresponding edge. It also supports continuous zooming with the mouse wheel, which provides an overview of several adjacent screens at the same time up to a complete bird's eye view of the virtual desktop (Figure 7).

In the current implementation of this zoomable desktop, clicking on a particular virtual screen from an overview initiates an animated transition that zooms into it. Note that all applications remain accessible while in overview mode: they can be moved between virtual screens, resized or closed by the user who can also interact with them as usual. We have also implemented a simple version of Apple's *Exposé*, which scales down the windows on the screen and tiles them to make them all visible. Like our zoomable desktop and as opposed to *Exposé*, this version lets the user interact with applications as usual. The zoomable desktop and our *Exposé*-like could probably be combined, the latter allowing users to access windows otherwise unreachable.

[7] in addition to the focus-plus-context approach, Scalable Fabric supports the notion of tasks (groups of windows) which we don't support in this configuration

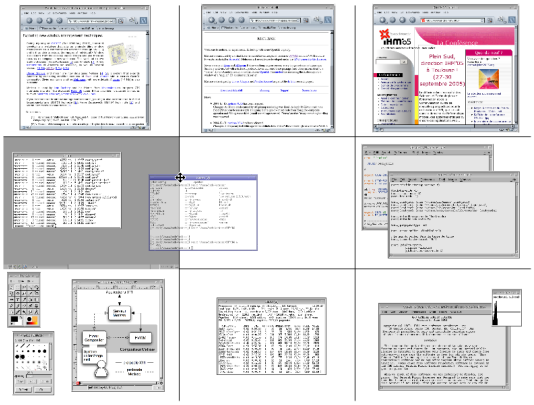


Figure 7: Bird's eye view of a virtual desktop made of nine virtual screens.

DISCUSSION AND DIRECTIONS FOR FUTURE WORK

The main problem we have when we demonstrate a Metisse-based desktop is that people tend to assume that Metisse is what they see (e.g. a 3D desktop). Metisse is not a 3D desktop! It is a highly tailorable, graphics-rich, out-of-the-box replacement for existing X desktops. Which makes it a perfect tool for rapid high-fidelity prototyping of window management techniques. As we already stated, we believe this is an important point and a great improvement over other experimental desktops because it should make it possible to conduct longitudinal studies of new window management techniques.

Although we could easily reproduce every detail of existing techniques such as Scalable Fabric or Exposé, our goal was rather to reproduce their most characteristic effects and to show that Metisse would make it easy to combine them. Again, the important point is that Metisse is a real system that can be used on a daily basis with any X Window application. The new interactions techniques that we have implemented may seem obvious to some HCI researchers (e.g. undo operations, continuous zooming, tabletop interaction). However, to our knowledge, they have never been available in a desktop environment with real, unmodified applications.

In a previous paper [24], we explained how basic features of modern graphics libraries (e.g. 3D transformations, alpha blending, texture mapping, lighting) could help us create innovative window management techniques. In this paper, we have described how Metisse supports their implementation. We strongly believe that a well chosen subset of these techniques could significantly improve window management tasks. We recognize that FVWM is far from ideal from an end-user development perspective. However, we believe that it is simple and powerful enough for researchers to configure the subset of Metisse capabilities to be used. We will probably investigate other ways for end-users to fine tune the resulting environment.

Another related problem that we face is that the usual user interface of a window manager provides only a few controls (e.g. a title bar, borders and several buttons). We are investigating ways of providing more control without adding more widgets or keyboard shortcuts. We have recently implemented *control menus* in FvwmCompositor, that users can trigger by clicking on the borders of a window. These menus support both the selection of operations from a circular menu and the control of their execution with a single gesture [20]. As an example, if the user clicks on the left border of a window and initiates a horizontal move, the menu starts an interactive resize operation. If the mouse moves up or down, it starts a scale or rotate operation.

Performance and preliminary evaluation

One of us uses Metisse on a regular basis. In particular, a large part of FvwmCompositor has been developed inside FvwmCompositor itself (modify the code, compile and restart FvwmCompositor!). Metisse works perfectly well for day to day activities such as e-mail and instant messaging, digital pictures and web browsing, text and image editing, as well as small-sized videos. Rotations and scaling are often used to reduce overlapping. The overview mode of the zoomable virtual desktop is used for rearranging windows. One limitation of the current implementation is that OpenGL applications cannot be hardware-accelerated in Metisse, which makes them slower than usual. The current way of dealing with this is to switch from Metisse to the native desktop for OpenGL applications (and high-resolution videos).

The computer used by this author is a two years old laptop running Linux, with a 2 GHz Pentium IV, 768 MB of memory and a Radeon Mobility M6 graphics card with 32 MB of memory. On that machine, applications making an extensive use of the X drawing API run at up to fifty frames per second and high-resolution videos can't be played at their nominal frame rate. As an example, a 720x576 Divx video is displayed at only twelve frames per second. Many applications can run together without any problem. The limited amount of video memory sometimes causes temporary performance problems. However, the iconification of a few windows is usually enough to free some memory and return to the standard performance level. Several tests with a more recent graphics card, a Nvidia GeForce with 128 MB of memory, doubled the frame rate of drawing-based applications and allowed to view the Divx video at nominal frame rate.

A preliminary version of the Metisse source code has been publicly released in June 2004. A few maintenance releases have followed. The last release has been downloaded more than 7000 times and the Metisse web site serves around 800 pages by day. About one hundred people have contacted us by e-mail, asking for support, giving some feedback and reporting a few bugs. Most people who contacted us were very positive and a few of them actually use Metisse as their default desktop. We are currently preparing an evaluation survey that will be distributed with the next release. We are also working on an extension of Metisse that will allow the recording of all window operations for later replay and analysis.

Towards a variety of compositors

The Metisse server is currently being used by a group of people from Mekensleep [8], a company developing an OpenGL-based on-line poker game. Their original motivation was to be able to integrate an external chat application in the game. Once they had implemented a basic Metisse compositor in their application, they realized that it could also be used to bring 2D interfaces built with traditional GUI toolkits such as GTK+ into their OpenGL scene (Figure 8).

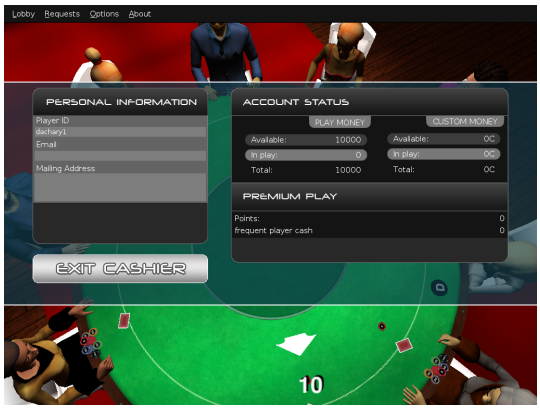


Figure 8: Poker3D as a Metisse compositor: windows containing GTK+ interface elements are rendered by the Metisse server and displayed by Poker3D on top of the 3D scene.

This idea of using Metisse to integrate 2D interfaces in 3D environments seems very interesting to us. We hope that Metisse will be used by other researchers in similar ways. As a consequence, we are currently developing a library to facilitate the use of the Metisse server and the implementation of new compositors. FvwmCompositor has a now relatively long history. Some code clean-up is being made, which will probably facilitate the implementation of new experimental window management techniques by other researchers.

As we said in the previous section, FvwmCompositor can display multiple instances of a window. But it can do more: it can duplicate a window region (as described in [28]), create holes in a window (as suggested in [13]), create a new window from pieces of others, and embed part of a window into another one (Figure 9). In fact, FvwmCompositor should be seen as a *window region* compositor. The next natural step is to move to a *widget compositor*. We are currently investigating the use of accessibility APIs to obtain the widget tree associated to a particular window and use it to support new interaction techniques. As an example, since FvwmCompositor already handles all user input, this should make it possible to use semantic pointing [5] for window management.

CONCLUSION

In this paper, we have described Metisse, a window system created to facilitate the design, the implementation and the evaluation of innovative window management techniques.

[8] http://www.mekensleep.com/

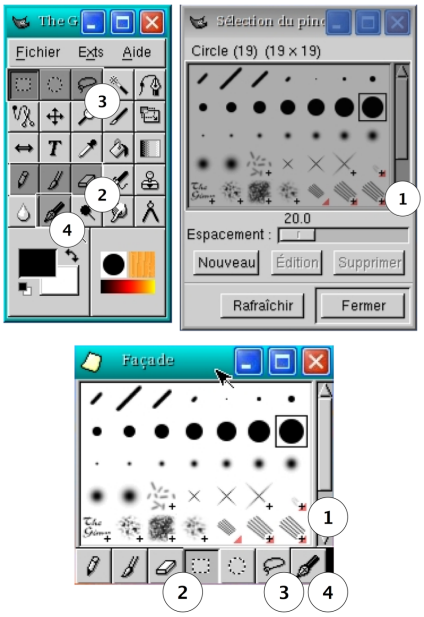


Figure 9: Assembling screen regions: the bottom window has been constructed by assembling four regions from the two windows above. See [27] for more details about the involved interactions.

We have presented the general architecture of the system and described some of its implementation details. We have presented several examples of uses that illustrate its potential and several directions for future research.

Metisse is available from http://insitu.lri.fr/metisse/

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their useful comments and suggestions about this work. Thanks to Wolfgang Stuerzlinger who challenged us with new interesting ideas. Thanks also to Loïc Dachary who carried Metisse with him when he moved to Mekensleep.

REFERENCES

1. Mac OS X v10.2 Technologies: Quartz Extreme and Quartz 2D. Apple Technology brief, October 2002.

2. M. Beaudouin-Lafon. Novel interaction techniques for overlapping windows. In *Proceedings of UIST 2001*, pages 153–154. ACM Press, November 2001.

3. J. Beda. Avalon Graphics: 2-D, 3-D, Imaging And Composition. Presentation at the Windows Hardware Engineering Conference, May 2004.

4. B. Bell and S. Feiner. Dynamic space management for user interfaces. In *Proceedings of UIST 2000*, pages 239–248. ACM Press, 2000.

5. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *Proceedings of CHI 2004*, pages 519–526. ACM Press, April 2004.

6. S.K. Card, M. Pavel, and J.E. Farrell. Window-based computer dialogues. In *Proceedings of Interact'84*, pages 239–243. North-Holland, September 1984.

7. B.-W. Chang and D. Ungar. Animation: from cartoons to the user interface. In *Proceedings of UIST 1993*, pages 45–55. ACM Press, 1993.

8. P. Dietz and D. Leigh. DiamondTouch: a multi-user touch technology. In *Proceedings of UIST 2001*, pages 219–226. ACM Press, 2001.

9. P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proceedings of UIST 2004*, pages 193–196. ACM Press, 2004.

10. J. Gettys and K. Packard. The (Re)Architecture of the X Window System. In *Proceedings of the 2004 Linux Symposium, Volume 1*, pages 227–237, July 2004.

11. D. A. Henderson and S. Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG)*, 5(3):211–243, 1986.

12. S.E. Hudson and J.T. Stasko. Animation support in a user interface toolkit: flexible, robust, and reusable abstractions. In *Proceedings of UIST 1993*, pages 57–67. ACM Press, 1993.

13. D. Hutchings and J. Stasko. Revisiting Display Space Management: Understanding Current Practice to Inform Next-generation Design. In *Proceedings of GI 2004*, pages 127–134. Canadian Human-Computer Communications Society, June 2004.

14. D. Hutchings and J. Stasko. Shrinking window operations for expanding display space. In *Proceedings of AVI'04*, pages 350–353. ACM Press, 2004.

15. D. Hutchings and J. Stasko. mudibo: Multiple dialog boxes for multiple monitors. In *Extended abstracts of CHI 2005*. ACM Press, April 2005.

16. E. Kandogan and B. Shneiderman. Elastic Windows: evaluation of multi-window operations. In *Proceedings of CHI 1997*, pages 250–257. ACM Press, March 1997.

17. J. Lasseter. Principles of traditional animation applied to 3d computer animation. In *Proceedings of SIGGRAPH 1987*, pages 35–44. ACM Press, 1987.

18. B. Myers. A taxonomy of window manager user interfaces. *IEEE Computer Graphics and Applications*, 8(5):65–84, sept/oct 1988.

19. B. Myers. A brief history of human-computer interaction technology. *ACM interactions*, 5(2):44–54, march/april 1998.

20. Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus: excecution and control in a single interactor. In *Extended abstracts of CHI 2000*, pages 263–264. ACM Press, 2000.

21. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, Jan-Feb 1998.

22. G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. Hutchings, B. Meyers, D. Robbins, and G. Smith. Scalable Fabric: Flexible Task Management. In *Proceedings of AVI'04*, pages 85–89, May 2004.

23. 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. In *Proceedings of CHI 2000*, pages 494–501. ACM Press, April 2000.

24. N. Roussel. Ametista: a mini-toolkit for exploring new window management techniques. In *Proceedings of CLIHC 2003*, pages 117–124. ACM Press, August 2003.

25. R.W. Scheifler and J. Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, 1986.

26. D. Smith, A. Raab, D. Reed, and A. Kay. Croquet: A menagerie of new user interfaces. In *Proceedings of C5 2004*, pages 4–11. IEEE Computer Society, January 2004.

27. W. Stuerzlinger, O. Chapuis, and N. Roussel. User interface façades : towards fully adaptable user interfaces. Rapport de Recherche 1408, LRI, Université Paris-Sud, France, April 2005. 9 pages.

28. D. Tan, B. Meyers, and M. Czerwinski. Wincuts: manipulating arbitrary window regions for more effective use of screen space. In *Extended abstracts of CHI 2004*, pages 1525–1528. ACM Press, 2004.

29. M. Tomitsch. Trends and Evolution of Window Interfaces. Diploma thesis, University of Technology, Vienna, December 2003. 132 pages.

30. M. van Dantzich, G. Robertson, and V. Ghorokhovsky. Application Redirection: Hosting Windows Applications in 3D. In *Proceedings of NPIV99*, pages 87–91. ACM Press, 1999.

User Interface Façades: Towards Fully Adaptable User Interfaces

Wolfgang Stuerzlinger[†], *Olivier Chapuis*[*], *Dusty Phillips*[†] & *Nicolas Roussel*[*]

[†]Interactive Systems Research Group
Comp. Science & Engineering, York University
Toronto, Canada
wolfgang | dustyp @cse.yorku.ca

[*]LRI (Univ. Paris-Sud - CNRS) & INRIA Futurs [1]
Bâtiment 490, Université Paris-Sud
91405 Orsay Cedex, France
chapuis | roussel @lri.fr

ABSTRACT
User interfaces are becoming more and more complex. Adaptable and adaptive interfaces have been proposed to address this issue and previous studies have shown that users prefer interfaces that they can adapt to self-adjusting ones. However, most existing systems provide users with little support for adapting their interfaces. Interface customization techniques are still very primitive and usually constricted to particular applications. In this paper, we present User Interface Façades, a system that provides users with simple ways to adapt, reconfigure, and re-combine existing graphical interfaces, through the use of direct manipulation techniques. The paper describes the user's view of the system, provides some technical details, and presents several examples to illustrate its potential.

ACM Classification: H.5.2 [Information interfaces and presentation]: User interfaces - Graphical user interfaces.

General terms: Algorithms, Design, Human Factors.

Keywords: Adaptable user interfaces.

INTRODUCTION
User interfaces are becoming more and more complex as the underlying applications add more and more features. Although most people use only a small subset of the functionalities of a given program at any given time [19], most software make all commands available all the time, which significantly increases the amount of screen space dedicated to interface components such as menus, toolbars and palettes. This quickly becomes a problem, as users often want to maximize the space available for the artifacts they are working on (e.g. an image or a text document). One reason for this problem might be that most user interfaces are still designed by software programmers today, a fact that is only slowly changing. However, even trained interface designers cannot always foresee how a software package is going to be used in practice, especially if the package is used by a large variety of different users. This makes creating flexible user interfaces a major challenge.

Consider GIMP as an example. The latest version of this image manipulation program has 22 persistent *dialogs* for managing brushes, colors, fonts, etc. Although dialogs can be docked together in an arbitrary number of windows, this only increases the window management overhead and increases the average distance to the drawing tools & functions from the drawing area. Users adapt with various strategies, such as having all dialogs on a secondary monitor, or overlapping the drawing area with dialogs. On the other hand, some applications use an all-in-one window logic, which provides less flexibility in terms of user interface layout.

One way of dealing with the growing number of application features and the desire to optimize screen space is to allow users or applications to customize the user interface. These two concepts have been studied for some time by the community (e.g. [17, 18]). Today, they are most often referred to as *(user-)adaptable* and *adaptive* (or *self-adapting*) interfaces [19]. Adaptive interfaces change their appearance based on some algorithm, such as a least-recently used criterion. One recent example is the menus of the Microsoft Office suite. Adaptable interfaces, on the other hand, can be configured by the user to suit his or her own criteria. Many applications, for example, make it possible to interactively customize their toolbars with simple drag-and-drop operations.

Adaptive interfaces can exhibit some unpleasant side effects such as surprising the user by moving or removing menu entries. Previous studies have also shown a desire for the user to be able to control and override the automatic system whenever needed [11]. Adaptable interfaces suffer from the problem that new 'secondary' interfaces and interaction techniques must be added to support the customization of the 'primary' interface. A

UIST'06, October 15–18, 2006, Montreux, Switzerland..

[1]projet In Situ, Pôle Commun de Recherche en Informatique du plateau de Saclay

comparison of static, adaptive, and adaptable menus showed that users could optimize their performance if they knew about the possibility of adapting and were able to adapt their menus with a simple interface [8]. Another interesting finding is that the adaptable user interface did not perform worse than the other two alternatives. Furthermore, participants greatly preferred the adaptable interface to the two other alternatives, a fact that we see as strong motivation for additional research in this area.

While the idea of adding adaptation functionality to user interface toolkits seems attractive at first glance, it has the drawback that it will make the already complex APIs of these toolkits even more complex, requiring yet more code to be written by application programmers. This is clearly not a positive thing and would not speed adoption of the fundamental paradigm of adaptable interfaces. Moreover, modifying the toolkits would leave it to programmers or interface designers to decide what can be configured and how. Yet, again, these professionals cannot necessarily foresee all potential ways of adapting an application. Phrased differently, we believe that users should be in control of the adaptation process, not the original software authors.

In this paper, we present User Interface Façades, a system designed to address this issue. The rest of the paper is organized as follows. In the next section, we present an overview of previous work and motivate our research. After presenting the main ideas of User Interface Façades, we discuss how we implemented them. Then we present several examples to illustrate the concepts, followed by the conclusion.

MOTIVATION

Skins and *themes* are two of the simplest forms of user interface customization. The notion of a *skin* comes from video games such as Quake that allow players to alter the appearance of their character and has been adopted by many media players. *Themes* extend this notion by sharing a common visual style among different applications, as specified by the user at run time. A skin, or a theme, can simply consist of a set of colors or textures used by existing drawing code. It can also partially or completely replace that drawing code, possibly adding complex output modifications [7]. In addition to the visual style of interface elements, skins and themes can also specify the layout and to a lesser degree the behavior of these elements. Recent work has extended this approach to bridge the gap between appearance and semantic meaning [9, 6]. However, although these allow visual designers to customize interfaces using off-the-shelf drawing tools such as Adobe Photoshop or Illustrator, these systems remain out of reach for end-users who can only choose between predefined theme options.

One of the biggest obstacles for adaptable interfaces is that it requires a fairly substantial programming effort to add this capability to a software package. Most user interface toolkits offer no support for implementing adaptable interfaces. This factor has certainly hindered the adoption of the idea of adaptable interfaces. As a notable exception, Apple's Cocoa toolkit provides developers with a toolbar widget that users can customize at runtime using drag and drop operations. However, the customization interface is far from optimal, as it does not allow for undoing changes or reverting to previous versions and employs a fixed window, which is inconvenient in many situations. Microsoft Office applications also allow users to customize their various toolbars and menus. But again, the customization interface has a number of serious flaws (Figure 1).

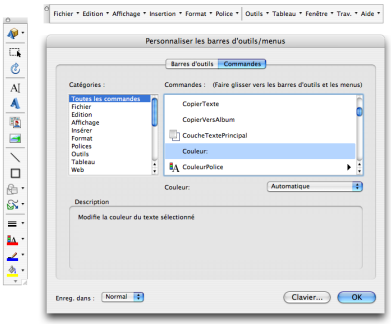


Figure 1: Microsoft Word 2004 interface for customizing menus and toolbars. The left list contains 22 command categories. The right list shows the commands relevant to the selected category. More than 1100 commands are available through this interface. They can be dragged to/from menus and toolbars, but these operations cannot be undone. Commands already in menus or toolbars still appear in the list. The window is about 600x500 pixels, can be moved, but not resized.

Bentley and Dourish [3] introduced an interesting distinction between *surface customization*, which allows users to choose between a predefined set of options, and *deep customization*, which allows them to customize deeper aspects of a system, such as integrating an external translation program with a word processor. They point out two problems that our above examples also illustrate. First, the level of customization provided by most systems lies above the functionality of the application, rather than within it. Second, these systems often require the learning of new languages to describe new behaviors.

Fujima et al. recently proposed the C3W system (Clip, Connect and Clone for the Web) to generate new HTML documents by cloning individual HTML elements from other documents and allowing for computation on these elements using a spreadsheet model [10]. While this approach supports deep customization, C3W is limited to Web technologies and does not allow the user to change or replace widgets nor to add new widgets to existing documents. Hutchings and Stasko proposed the more generic notion of *relevant window regions* and suggested to add the ability to create copies of these regions that could be manipulated as independent windows [14]. Tan et al. implemented this idea in their

WinCuts system [22]. However, this system is unable to merge several regions into a new window, which is clearly a limiting factor. Its implementation also has several problems that make it hardly usable on an everyday basis (e.g. it relies on periodic polling of window content, popup menus and dialog boxes appear on the source window, etc.). Berry et al. introduced a system that can selectively hide content based on the users' privileges via various forms of blurring [4]. Internally, this system works similarly to WinCuts.

Hutchings and Stasko also suggested allowing users to remove irrelevant parts of windows [14]. The same idea was mentioned in [21] and partially implemented (windows could be cropped to a set of pre-defined shapes). Finally, Hutchings and Stasko proposed to replicate dialog boxes on multiple monitor configurations until the user interacts with one of the copies [15]. In this same paper, they concluded that window operations like these should be implemented within the window manager rather than using a separate application.

Based on the above discussion, we formulated the following criteria for adaptable user interface:

- *Fast, simple, just-in-time customization*: Users should be able to adapt interfaces without advance planning, whenever needed, and should be able to do this in a fast and simple way, e.g. with direct manipulation techniques.
- *Not only global customizations, but also local ones*: Most adaptable interfaces only support global changes, which forces users to undo them at some point. Global/local can be interpreted in different ways (e.g. persistent/temporary, all documents/this document). Users should be able to specify the scope of interface customizations. It should be possible, for example, to customize the toolbars of an application for a specific session only, or even for a specific document.
- *Deep customization*: Users should not be restricted to a set of pre-defined options but should be able to define new ones. Again, 'set of options' can be interpreted in different ways, e.g. a tool set or a set of specific locations where tools can be placed. Users should be able to select anything on the screen, change the way it operates (not only visual appearance), cut it out, duplicate it, or replace it with something else. The latter should be done in a manner that removes the 'old' user interface, or at least makes it invisible.
- *Cross-application customization*: Interface customizations should make it possible to combine or link together different applications.

USER INTERFACE FAÇADES

This work focuses on applications with a graphical user interface, as opposed to command-line systems. We are more specifically interested in applications where the interaction focus is a single or, at best, a few document(s). In such applications a large work area dominates the main window, with user interface elements clustered around. Examples include drawing packages, text processors, spreadsheets, etc.

A user interface *façade* is a user-specified set of graphical interfaces and interaction techniques that can be used to customize the interaction with existing, unmodified applications. This section provides a general overview of how users interact with such façades. Implementation details and more specific usage scenarios follow in the next two sections.

Copying and pasting screen regions

A basic functionality of the Façades system is the ability to copy interface components from one window to another while maintaining a one-to-one functional relationship between the copy and the original. Using the mouse and a specific modifier key the user can select one or more rectangular *source regions*. A drag operation on these regions duplicates them. Dropping the duplicates on the desktop puts them in a new *façade window*. Façade window creation from source regions is also accessible through a menu that pops up when one clicks on one of the regions using the right mouse button. A new command also makes it possible to clone a complete window through its standard window menu.

Dropping duplicated interface components onto the side of a façade window automatically expands the façade to make room for the new duplicate at this side. Dropping components into free space inside a façade window simply adds it in that space. Duplicates can also be dropped on any existing window, and will overlay the dropped component over the existing content. Figure 2 shows a user incrementally constructing a façade window by selecting widgets from three dialogs of the GIMP application. The scenario here is that the user wants to optimize the interface by packaging frequently used tools in an ad-hoc way, rather than using the GIMP developers' pre-packaged toolsets. The upper row of images shows four selected regions in two GIMP dialogs (displayed as semi-transparent rectangles) and the resulting façade window, which contains the duplicated regions. The lower row illustrates the addition of a fifth duplicated component to this window.

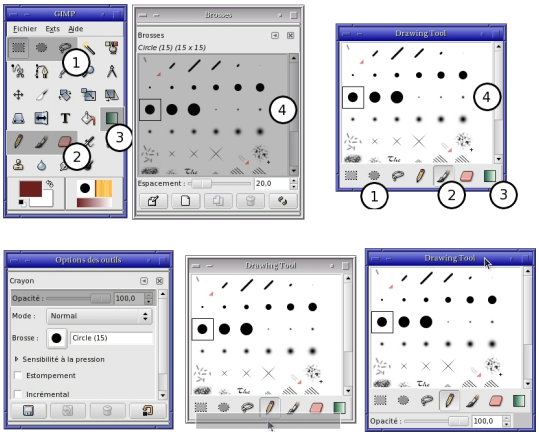


Figure 2: Creating a façade from several GIMP dialogs.

The same source region can be used in several façades (i.e. it can be duplicated several times), and a façade can contain an arbitrary number of duplicates. After a façade has been created, the user typically hides or iconifies the source window(s) and the system transparently passes mouse movements and clicks over the façade to the appropriate source region. Conversely, source region updates are replicated in their corresponding duplicates. Overlay windows such as popup menus are correctly handled when triggered from a duplicate. The system also transparently manages the focus and stacking order according to standard window manager rules. In effect, the behavior of a duplicate is indistinguishable from the original source region to the user.

Parts of the above ideas have been previously presented by Tan et al. [22] (e.g. the ability to duplicate multiple screen regions into individual windows) and Hutchings and Stasko [14, 15] (e.g. the ability to duplicate windows). However, the ability to create new windows that seamlessly combine multiple screen regions and the ability to paste regions over arbitrary windows are unique to our work.

Cutting screen regions

In addition to supporting the creation of façade windows, the system also allows users to create holes in windows, via a context-sensitive menu that becomes active after a region on the screen has been selected. This can be used to remove uninteresting parts or to reveal other windows beneath. As an example, consider revealing a small utility, such as a calculator or calendar, inside an unused region of a primary application (Figure 3). As the keyboard focus follows the mouse position in Façades, the user can then simply interact via the keyboard with the partially covered calculator 'through' the hole. This is especially interesting if the primary application is run in full-screen mode, which is something that traditional window systems do not support. Holes created in a window with the Façades system can be deleted via a command in the window menu or with a keyboard shortcut.

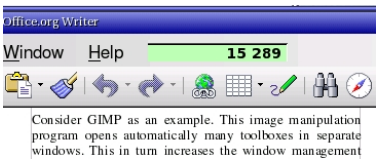


Figure 3: Accessing a calculator through a hole in a word processor.

Using external components to interact with applications

One idea that appears rarely in the discussion about adaptable user interfaces in the literature is that the user cannot only adapt the visual appearance of the interface, but also the *interaction* part of it. Façades allows the user to do this without any change to the code of the underlying application. One possible modification is to replace a component of a GUI with another GUI component, typically created by a third party. For example, with Façades the user can replace a dropdown list widget containing all countries of the world with a map widget or alternatively some radio buttons for the small set of countries that the user needs frequently in his or her work. Another modification allows the user to modify the interaction with standard components. For example, the user can integrate scrolling and zooming by remapping how mouse movements on a standard scroll bar are interpreted. These and other examples will be discussed in more detail later in the paper.

Managing Façades

To enable the quick recall of a façade, the user can give it a name and save it through a specific command in the window menu. When saving, the user can set options in a dialog: automatic recall, automatic hiding of source windows at recall time and the use of the window title in the saved description of the façade.

At a later time and if all relevant windows are open, the system can then recreate a façade automatically, or on user demand via the window menu. For this, Façades monitors all window related events and identifies matching configurations via window geometry, class, and resource names. If applicable, replacement widgets are automatically instantiated. A sub-menu of the normal desktop menu also contains a list of all saved façades for all currently active window configurations.

Contributions

In summary, we present the following new techniques for adaptable user interfaces:

- Seamlessly merge duplicated screen regions into new windows enabling the creation of new user interfaces for existing applications.
- The ability to create holes in windows and to seamlessly overlay duplicated content over existing windows.
- The ability to seamlessly replace widgets with other (potentially customized) widgets.
- The ability to seamlessly change the interaction with widgets, including the composition of widget behaviors, as well as the creation of toolglasses and other advanced user interface techniques.
- Implementing all of the above in a way that does not require any coding, with a simple-to-use interface based on drag-and-drop.

The following implementation section provides the technical details that make the system efficient and reliable and discusses related issues such as resizing.

IMPLEMENTATION DETAILS

In this section we describe how we implemented Façades and how it is integrated into a windowing system. Conceptually, Façades acts as a transparent layer over the window system that redirects input events and duplicates window regions as specified by the contents of each façade window. For seamless duplication it uses the offscreen buffer capabilities of Metisse [5], as well as its input redirection facilities. Façades determines widget

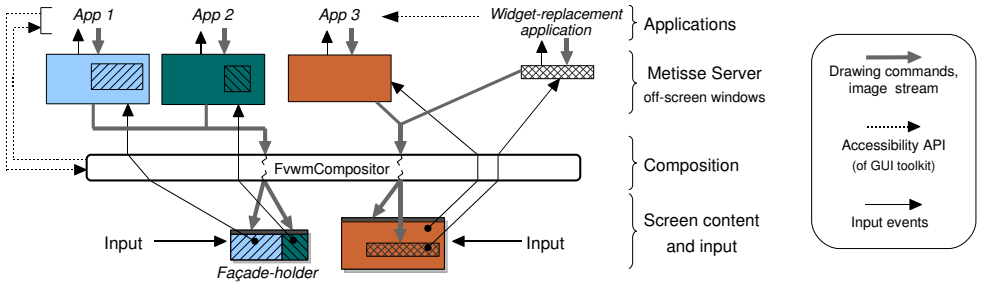


Figure 4: Illustration of input event and image flow in Façades system

positions through the accessibility API of modern GUI toolkits. Finally, widget replacement and interaction modification is achieved via the instantiation of simple replacement applications that are again based on accessibility API calls. Figure 4 illustrates how the various components of Façades work together. The left hand part shows a façade that composites two separate windows, whereas the façade for 'App 3' utilizes widget replacement. In the following subsections we first discuss how input & output are redirected and then mention how we access and replace widgets.

Basic input/output management using Metisse

Façades is implemented based on Metisse [5]. The Metisse architecture uses a compositing approach, making a clear distinction between window rendering and the interactive compositing process. The *Metisse server*, an enhanced X Window server, renders applications offscreen. In Façades, window images are composited by a separate application, *FvwmCompositor*, which is based on the window manager FVWM. Mouse and keyboard events received by FvwmCompositor are usually sent to appropriate applications through the Metisse server. In some cases, however, events are directly handled by FvwmCompositor itself, e.g. to implement façade region selection and window management commands, such as 'Alt-F4'. Specific façade commands in FvwmCompositor are accessible from FVWM to enable the creation of façade windows, holes, etc. Conversely, FvwmCompositor uses FVWM commands to handle pop up menus or to indicate the real mouse focus when the pointer is over a duplicate.

Each façade window is managed by an instance of a simple program, *façade-holder*, that keeps track of the duplicate regions it contains and creates a new X window to hold them (duplicates are then displayed by FvwmCompositor in that window). This program is invoked each time one or more duplicates are dragged from a source window and dropped onto the desktop. Each duplicate is described in *façade-holder* by a tuple of the following form: $(XID, src_x, src_y, src_{width}, src_{height}, dst_x, dst_y)$ where XID identifies the source window, $(src_x, src_y, src_{width}, src_{height})$ specifies the original region geometry relative to the source window, and (dst_x, dst_y) specifies its position in the façade window.

Façade-holders publish these tuples to other programs, including FvwmCompositor, through an X atom[1]. When a new duplicate is pasted into an existing façade window, FvwmCompositor sends an X client message with the source information for the duplicate to the *façade-holder*. Upon receiving this message, the *façade-holder* computes the local geometry of all its elements and updates its atom accordingly. FvwmCompositor catches this new layout and redraws the façade window.

FvwmCompositor maintains a list of duplicated regions for every window and handles updates for every content change. It also handles the necessary focus changes as the mouse moves from one duplicated region to another. Mouse and keyboard events for a façade window are normally sent to the appropriate source window. Similarly, clicking on a duplicate region raises the façade window, not the corresponding source window. FVWM handles these situations by distinguishing two types of focus: one for window management tasks, and the other for interacting with window content.

Transient overlay windows, such as popup menus or tooltips, are rendered in the right place. When such a window is mapped, FvwmCompositor computes its 'parent' window, i.e. the source window that is most probably responsible for this new window to appear. If the mouse pointer is over an element of the parent, FvwmCompositor positions the overlay based on the parent location and the element position and geometry. If the parent window is invisible, the overlay window is placed close to the pointer. Transient dialogs are placed so that their center is aligned with their façade window.

Iconification of source windows also poses specific problems. The usual way of iconifying X windows is to 'unmap' them in the server and replace them with a new graphical object. But unmapped windows do not get redrawn and cannot receive events. Consequently, when a source window is iconified in Façades, it is not unmapped, treated as iconified by FVWM and not rendered by FvwmCompositor. When a source window is closed, FvwmCompositor notifies the corresponding façades by sending them an X client message that specifies the region(s) to be removed. When its last element is removed, a façade either remains empty on-screen for

[1]Atoms are an X Window specific publish/subscribe mechanism

later use, or is automatically destroyed in the case of cloned windows. Façade and cloned windows are not resizable by the user. Cloned windows are automatically resized to match the geometry of their source window. Duplicated regions are kept visible in façades only if they are still visible in their source window.

All menus to manage façades are handled by FVWM. Some are statically defined in configuration files. Others are dynamically created by FvwmCompositor (e.g. the list of previously saved façades for a window). Saving a façade generates a human-readable description of its elements on disk. FvwmCompositor uses the geometry, class, resource names, and optionally the title of the source windows of a façade to create a heuristically-unique identifier. Widget-related information obtained from accessibility APIs can also be used to make this identifier more robust. FvwmCompositor loads all saved façade descriptions at startup and whenever windows are created or resized, it checks for matching façade descriptions and creates them accordingly.

Taking advantage of accessibility services

Widget-related information is very useful for creating façades. Knowing the position, size, type, and current state of each widget as well as having access to its actions offers a number of interesting possibilities. As an example, knowing the boundaries for each widget can facilitate the selection of widgets via snapping. There are several ways to obtain widget-related information and control widgets from the outside. In the current implementation of Façades, we use the accessibility APIs supported by most modern GUI toolkits.

Apple defined Universal Access APIs for its Carbon and Cocoa toolkits, Microsoft the Microsoft Active Accessibility & System.Windows.Automation frameworks, and X Window the Assistive Technology Service Provider Interface (AT-SPI), a toolkit-neutral way of providing accessibility services supported by GTK+, Java/Swing, the Mozilla suite, StarOffice/OpenOffice.org and Qt. All of these APIs can query the current position, size, type, and state of all widgets of an application. Furthermore, all possible widget actions can be activated via these APIs (e.g. one can cause selection events, trigger buttons, etc.). The following pseudo-code segment illustrates this for the example shown in Figure 9 via the AT-SPI accessibility API.

```
# Event handler for click at (x,y) on map
# Input: x, y, app_name (application name),
#        comp_name (widget name), comp_type (widget type)

# Map click to combobox list index
index = get_province_for_point(x, y)

# recursively find the accessible component in widget tree
application = desktop.find_app(app_name)
comp = application.find_component(comp_name, comp_type)
# get accessible action interface object
selector = comp.queryInterface("Accessibility/Selection")

# "aaaaand: Action!": fire event to widget
selector.selectChild(index)
```

For resizing there are two issues to consider. Any GUI application may resize it's window or widgets at any time or the user can resize the façade window itself. While the Façades system can detect the first kind of resize events via the accessibility API, any *automatic* change to a façade might break the layout of the façade as constructed by the user. This is clearly undesirable. Hence, we currently warn the user in this case and require that he/she fixes the problem manually. Second, a user can actively resize a façade window. While we could search for widgets that are resizable and try to adapt the layout accordingly, this would require an easy-to-use interface for specifying widget layout. As current layout methods typically have (too) many options, this is a research topic of it's own. Hence, we currently choose to disallow resizing of façades.

Other possible implementations

We have implemented the Façades system using Metisse and the accessibility API. The Metisse compositing architecture permits dynamic rendering of interface elements and handles input redirection. Furthermore, the FvwmCompositor interprets window management activities directly, while it passes interaction with façade content to the original applications.

It should be possible to implement Façades on other systems since accessibility APIs are now widely available. Moreover, the compositing approach is available under Mac OS X, Windows Vista and X Windows. However, neither OS X nor Vista have APIs flexible enough to freely redirect rendering output. For this reason WinCuts [22], called the PrintWindow function every second to update cut contents. In X Windows the full rendering API is accessible and documented. Even though this API is very complex (compared to Metisse) it seems possible to implement the rendering redirection part of Façades with it.

For input redirection, Mac OS X and Windows have no public API. As a workaround, WinCuts [22] draws a cursor over the interface elements, and the source window is kept in front of the true cursor. X Window has the X Event Interception Extension (XEvIE), but this extension is not powerful enough. For example it is not possible to send pointer events to a window, which is covered by another. A future X extension [20] may provide enough control of input redirection to implement something similar to Façades.

There are several other alternatives to extract widget related information and to activate widgets. For non-accessible GUI toolkits, one can extract information about widgets by modifying the dynamically linked toolkit library and adding functionality that returns (part of) the current widget hierarchy state on demand. Interaction with non-accessible widgets can be simulated via appropriate mouse and keyboard input events on the appropriate areas of a widget. E.g. to enter a particular string into a text-field, the system selects the field via a simulated mouse click, selects all old text and erases it via appropriate key sequences, and then

simulates entry of the new string. Most other widgets can be controlled with similar strategies. However, this is only a temporary workaround, as most GUI toolkits have already or are being retrofitted with an accessibility API, due to the strong need to add accessibility to all applications.

Alternatively, we can implement Façades via an intermediate layer in a window system. Such intermediate layers already exist today, e.g. in the form of user interface description languages (UIDL's). These are used to describe the user interface and how it activates the functionality of the application. XUL and XAML are two recent examples. If this intermediate layer is accessible from the outside, it is possible to implement Façades as an 'UIDL filter', which selectively replaces or duplicates widgets in the UIDL stream and adapts the calls to the application as appropriate.

DETAILED EXAMPLES / USAGE SCENARIOS

In the following section we present several examples of useful façades and explain how they were created.

Widget duplication

One application of Façades is to change the UI of a software package designed for right-handed people into a left-handed version, e.g. by moving the scrollbar from the right to the left-hand side. Another interesting idea is to duplicate a toolbar on two sides of the work area (or even on all four sides), which has the potential to significantly decrease average tool selection time. Figure 5 shows a file browser - Konqueror - with an additional toolbar at the bottom.

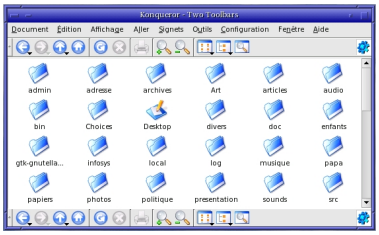


Figure 5: File browser with duplicated toolbar (bottom).

Façades also support the full duplication of whole windows, similar to [14, 15]. This functionality is activated via a titlebar menu. Duplication can be extremely useful in a multiple monitors setting, as it allows the user e.g. to duplicate the task bar or a panel with launch buttons on every monitor (with changes visible everywhere simultaneously). Another application of this idea is best illustrated with an example: Alice has two monitors on her desk, a laptop monitor, and an external monitor, which can be turned in any direction. Paul arrives in Alice's office and sits down on the other side of the desk. Alice turns the external monitor so that it faces Paul and duplicates her web browser onto the external monitor. Alice can then freely show her work while Paul is able to observe the demonstration.

Another example is the duplication of the GIMP toolbox window: toolboxes can be duplicated for each drawing window. We can even have two toolbox windows on each side of a drawing window to accelerate access to tools. Figure 6 illustrates such a layout.

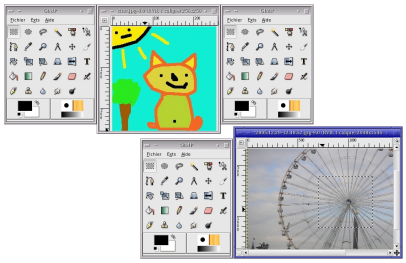


Figure 6: GIMP screen layout with duplicated toolboxes.

Another application of Façades is to duplicate useful notification areas into the area of an arbitrary window. As an example, consider duplicating the taskbar clock into the title bar or another unused area of a window (Figure 7). This is clearly interesting for full-screen applications and also for multi-monitor setups.

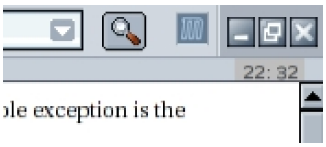


Figure 7: Duplication of taskbar clock into an unused area of Mozilla (near top right).

Widget duplication can also be used for the control of applications on secondary display devices. The main issue here is the reduction of mouse travel across large distances. We describe a two-monitor scenario that significantly extends an example from a technical report of Hutchings and Stasko [14]. Paul is a web developer and he edits a web page on his main monitor. On his secondary monitor he runs two different web browsers to test his work in real time. For this Paul first creates a façade consisting of the two reload buttons and the two vertical scrollbars of the browsers. Then he places this façade on his main monitor just to the right of the web editor. This allows Paul to quickly test his design by interacting with the façade and has the advantage that his mouse never needs to leave the main monitor.

We already presented an example of the power of combining elements above. Another example is the creation of a notification façade from different applications. Most e-mail programs display the 'inbox' as a list of one-line items containing information on the sender, subject, etc. Selecting (part of) this list and the two last lines of an instant messaging (IM) application allows the user to compose a novel 'contact' notifier façade. The advantage of such a notification application compared to the

usual small notificators in the taskbar is that it gives *simultaneously* information on new mails *and* new IM messages including the sender name. Users can then use this information to decide whether to switch from their current work to answer a message. Moreover, the user can even answer an e-mail message without switching to the full mail reader window as he/she can right-click on an e-mail's header line. One disadvantage of such a notification window is that it uses more screen space than the rather minimal taskbar notificators. However, Metisse has the ability to scale windows. Hence, such notifications can be also scaled (e.g. by reducing by 30%, which still maintains readability).

Widget replacement

Façades also targets the replacement of standard GUI widgets with other widgets. Consider a scenario where a user frequently uses a few options in a long list widget and only rarely uses other entries. A classical example is a call-center where data about each incident is recorded, and where the client base consists of many users in a small set of countries, but also a few others from around the world. Instead of having to choose every time from the list of all countries on the planet in the incident-entry form, it is much more efficient to have quick access to the subset of frequently used countries and provide a separate way to access the full list. As the call-center software developer cannot foresee which countries will be used frequently and how large that set will be, it is advantageous to give the user control of this GUI aspect. Figure 8 depicts an address entry form application for specifying addresses in Canada, the dialog that lets the user specify the provinces that appear in the façade, and the façade itself.

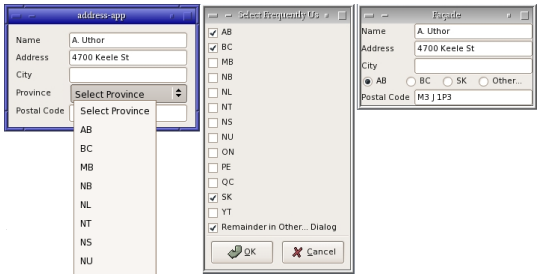


Figure 8: Original address entry application (left), the façade construction dialog, which lets the user select frequently used entries (middle) and the final façade for the application with a set of radio buttons (right).

In Façades, a user can access this functionality by first selecting a widget, then accessing a context-sensitive menu and selecting the appropriate entry. This will show a façade creation dialog with appropriate options. Once the user confirms their choice, Façades creates the custom replacement widget, which can be placed into a façade. The following pseudo-code illustrates the main parts of a generic combobox replacement widget. Code related to the dialogs for façade construction and 'Other...' functionality is not shown for brevity.

```
function combo2radio(app_name, combo_name):
  app = desktop.find_app(app_name)
  combo = app.find_component(comp_name, "combo box")
  # show dialog to user and return selected entries on close
  selection = SelectFromDialog(combo.items)
  # create a new window with the selected radio buttons
  radiobox = Window()
  for item in selection:
    radio = RadioButton(item)
    radio.bind("toggle", selectCallback, item.id)
    radiobox.add(radio)
  radiobox.display()

function selectCallback(widget, id):
  selector = widget.queryInterface("Accessibility/Selection")
  selector.selectChild(id)
```

Another option is to replace the provinces combobox in Figure 8 with an interactive map that allows direct selection of provinces in a map of Canada (see Figure 9). This is achieved via a replacement widget that maps click locations to selection events on the combo box. While this replacement widget is not as generic as the one depicted in Figure 8, it offers a better visualization, which some users may find easier to use. Depending on the user's needs, he or she may prefer one alternative or the other.

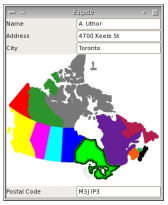


Figure 9: An alternative façade for the application from Figure 8.

As a different example for the replacement of standard widgets, consider a text-area widget and its enhanced replacement that adds syntax highlighting to make the contents easier to comprehend. With this replacement widget the user interface of *any* application with un-enhanced text-area widgets can be improved via Façades.

Similar to the shown examples, one can imagine many other replacement widgets and the code behind them will follow the general structure of the pseudo-code shown above, but tailored to the specifics of each pair of source and replacement widget. Consider e.g. enhancing an existing date entry field with an automatic pop-up calendar widget, whenever it is selected. Note however, that not all potentially possible widget replacements are 'good' from a UI designer standpoint, but this topic is beyond of the scope of this paper.

Interaction composition

Previous research has shown that toolglasses can improve user performance [16]. They are transparent UI elements, whose position is controlled by the non-dominant hand. The user then 'clicks-through' the desired mode-icon of the toolglass with the dominant hand to activate a function at the current cursor location. In Façades, the user can associate another (extended) input devices to a selected window or façade via the Façade window menu to create a toolglass. This causes the window to become semi-transparent and to remain always

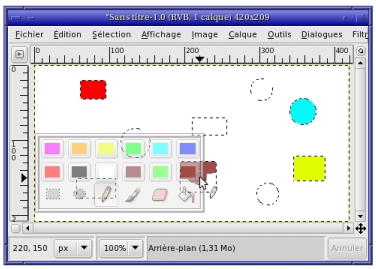


Figure 10: Using a palette façade as a toolglass.

on top over normal windows. The second input device, typically held in the nondominant hand, then controls the position of the toolglass window. Whenever the user presses a button on the device in the dominant hand, a click is sent to the toolglass (to activate the correct mode) and the press as well as subsequent drag events are sent to the window under the toolglass. This allows, for example, positioning the toolglass over the drawing area to select a tool and at the same time to start using that tool. For an illustration see Figure 10.

Moreover, a right click with the non-dominant device allows toggling between toolglass and normal window mode. This mode permits users to change tools with their non-dominant hand and to work with the selected tool with the dominant hand. We follow here the toolglass implementation of the post-WIMP graphical application CPN2000 [2]. One of the attractive features of Façades is that no change to the underlying application is necessary to fundamentally improve the user interface via toolglasses.

The OrthoZoom technique [1] combines scrolling and zooming. Mouse movement along the scrollbar direction results in scrolling, while orthogonal movements result in zooming. In addition, when the user releases the mouse button the original zoom factor is reestablished. This technique has been proven to be efficient and allows simultaneous control of scroll and zoom with a mouse. As a variation, one can map movement in the orthogonal direction to modulation of scrolling speed: the further the cursor is from the scrollbar, the slower the speed. This allows for very fine position control. Moreover, the two techniques can be combined: one orthogonal direction (e.g. left) is mapped to OrthoZoom and the other direction (e.g. right) modulates scrolling speed.

We have implemented these techniques for any accessible application. When the user right clicks with the façade modifier on a window, the system goes thought the widget tree and checks if the widget under the cursor has an accessible value. If yes, entries for scrolling are made available in the Façades menu. Additionally, if accessible zoom actions are available, OrthoZoom is made available, too. During interaction, the system then captures all events on the scrollbar and controls the application via the accessibility API. Clearly, the fluidity of the OrthoZoom techniques depends on the ability of the applications to rapidly zoom in and out, but this issue is beyond the scope of this paper. One unexpected benefit is that with this idea *any* widget with an accessible value can become scrollable - even a value text box can be controlled via this technique.

DISCUSSION

The current implementation of Metisse and the Façades system is fast enough to duplicate a 1590x860 video window at 25Hz on a recent laptop. Due to their complexity, accessibility APIs take some time to understand. However, once this has been mastered, replacement widget applications are very easy to generate. Modifying the interaction at the event level (e.g. remapping the action associated with a right click on a canvas), is also reasonably easy. The accessibility APIs provide all necessary data for Façades, but better access to graphical widget information could simplify some issues.

The ability to snap the selection to widgets is arguably the first thing that users notice positively about Façades. However, once users get used to the idea of freely adapting user interfaces of existing applications, they quickly come up with novel uses. One user, who uses a graphical editor in combination with command-line tools, has created a replacement widget with a "Save all" button that he places adjacent to the terminal window. The functionality behind the button activates the save function for all open editor windows to deal with the common problem of forgetting to save changes.

Another application of Façades is to monitor a larger set of mailboxes. As the user is waiting for different kinds of messages at different times, he creates a façade that monitors only those that are currently "interesting" and adapts that façade on demand to changed requirements. Yet another good use of Façades is to fix problems with suboptimally designed user interfaces. The search box of Thunderbird, for example, has a (barely visible) drop-down menu that allows changing between searching the subject, sender, and/or body. With Façades one can create a set of radio-buttons adjacent to the search box to make it easier to select the desired functionality.

Finally, Façades has the ability to make even static visualizations interactive by mapping mouse actions in certain regions to activations of other widgets, which is yet another way to enhance existing GUI's. However, we have to point out that not all modifications possible via Façades will improve the usability of a user interface. This is the trade-off faced by any user of a general-purpose tool.

CONCLUSION

In this paper, we presented a new approach to adaptable user interfaces. User Interface Façades allow end-users to quickly, flexibly and seamlessly change the interface of any application without coding. The system supports cutting, copying and pasting of screen regions, combined with the facility to overlay screen regions over other windows. We have shown how this approach supports both ad-hoc opportunistic customizations as well as persis-

tent ones. Furthermore, we demonstrated that Façades also supports deep customizations, such as the modification of the interactive behavior of arbitrary applications, something that previous work has not supported. We also presented several examples that demonstrate and extend the basic concept in several interesting directions (e.g. window management, multiple monitors, cross-application customizations, new scrolling techniques).

From a global perspective, we believe that Façades offers a good complement to direct programming of user interfaces. From the user's view, it greatly increases the flexibility of any GUI. From the programmers view, it is transparent, as no programming is required to give the user the ability to change the user interface. In the future, appropriate APIs to the Façades system may even enhance the interface programmer's or designer's ability to create good user interfaces.

The generalization from rectangular regions to more arbitrary regions is fairly simple from a high-level point of view and may increase the utility of façades even further. For future work, we plan to explore the Façades concept further and investigate how it can be integrate with UI description languages such as XUL & XAML. Furthermore, we will evaluate the adaptation facilities of Façades with user studies, similar to [8, 12].

In this context it is interesting to realize that User Interface Façades extend Apple's vision of the window system as 'a digital image compositor' [13]. More precisely, we can say that the addition of Façades to the standard window management and user interface paradigms allows us to put forth the vision of the window system as a fine-grained interactive graphical component compositor.

ACKNOWLEDGMENTS

Many thanks to the reviewers for their insightful comments, which led us to improve the paper. The initial part of this research was performed while the first author was on a sabbatical stay at In Situ, and Michel Beaudouin-Lafon's support is gratefully acknowledged. This work has been partially funded by NSERC and the French *ACI Masses de données* (Micromégas project).

REFERENCES

1. C. Appert and J.D. Fekete. Orthozoom scroller: 1d multi-scale navigation. In *Proceedings of CHI '06*, pages 21–30. ACM Press, 2006.
2. M. Beaudouin-Lafon and H. M. Lassen. The architecture and implementation of cpn2000, a post-wimp graphical application. In *Proceedings of UIST '00*, pages 181–190. ACM Press, 2000.
3. R. Bentley and P. Dourish. Medium versus mechanism: Supporting collaboration through customization. In *Proceedings of ECSCW '95*, pages 133–148. Kluwer Academic, September 1995.
4. L. Berry, L. Bartram, and K.S. Booth. Role-based control of shared application views. In *Proceedings of UIST '05*, pages 23–32. ACM Press, 2005.
5. O. Chapuis and N. Roussel. Metisse is not 3D desktop! In *Proceedings of UIST '05*, pages 13–22. ACM Press, October 2005.
6. S. Chatty, S. Sire, J.L. Vinot, P. Lecoanet, A. Lemort, and C. Mertz. Revisiting visual interface programming: creating gui tools for designers and programmers. In *Proceedings of UIST '04*, pages 267–276. ACM Press, 2004.
7. K. Edwards, S.E. Hudson, J. Marinacci, R. Rodenstein, T. Rodriguez, and I. Smith. Systematic output modification in a 2d user interface toolkit. In *Proceedings of UIST '97*, pages 151–158. ACM Press, 1997.
8. L. Findlater and J. McGrenere. A comparison of static, adaptive, and adaptable menus. In *Proceedings of CHI '04*, pages 89–96. ACM Press, 2004.
9. J. Fogarty, J. Forlizzi, and S.E. Hudson. Specifying behavior and semantic meaning in an unmodified layered drawing package. In *Proceedings of UIST '02*, pages 61–70. ACM Press, 2002.
10. J. Fujima, A. Lunzer, K. Hornbæk, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. In *Proceedings of UIST '04*, pages 175–184. ACM Press, 2004.
11. D. Funke, J. Neal, and R. Paul. An approach to intelligent automated window management. *IJMMS*, 38(6):949–983, 1993.
12. K.Z. Gajos, M. Czerwinski, D.S. Tan, and D.S. Weld. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of AVI '06*, pages 201–208. ACM Press, 2006.
13. P. Graffagnino. OpenGL and Quartz Extreme. Presentation at SIGGRAPH OpenGL BOF, Apple, 2002.
14. D. Hutchings and J. Stasko. An Interview-based Study of Display Space Management. Technical Report 03-17, GIT-GVU, May 2003.
15. D. Hutchings and J. Stasko. mudibo: Multiple dialog boxes for multiple monitors. In *Extended abstracts of CHI '05*, pages 1471–1474. ACM Press, April 2005.
16. P. Kabbash, W. Buxton, and A. Sellen. Two-handed input in a compound task. In *Proceedings of CHI '94*, pages 417–423. ACM Press, 1994.
17. E. Kantorowitz and O. Sudarsky. The adaptable user interface. *CACM*, 32(11):1352–1358, 1989.
18. T. Kühme. A user-centered approach to adaptive interfaces. In *Proceedings of IUI '93*, pages 243–245. ACM Press, 1993.
19. J. McGrenere, R.M. Baecker, and K.S. Booth. An evaluation of a multiple interface design solution for bloated software. In *Proceedings of CHI '02*, pages 164–170. ACM Press, 2002.
20. K. Packard. Coordinate transform redirection for composited window environments. Unpublished talk, FOSDEM 2006, Brussels (Belgium), 2006.
21. N. Roussel. Ametista: a mini-toolkit for exploring new window management techniques. In *Proceedings of CLIHC '03*, pages 117–124. ACM Press, August 2003.
22. D. Tan, B. Meyers, and M. Czerwinski. Wincuts: manipulating arbitrary window regions for more effective use of screen space. In *Extended abstracts of CHI '04*, pages 1525–1528. ACM Press, 2004.

Copy-and-Paste Between Overlapping Windows

Olivier Chapuis[1,2] **Nicolas Roussel**[1,2]
{chapuis,roussel}@lri.fr

[1]LRI - Univ. Paris-Sud & CNRS
Bât. 490, F-91405 Orsay, France

[2]INRIA
Bât. 490, F-91405 Orsay, France

ABSTRACT
Copy-and-paste, one of the fundamental operations of modern user interfaces, can be performed through various means (e.g. using the keyboard, mouse-based direct manipulation or menus). When users copy and paste between two different windows, the process is complicated by window management tasks. In this paper, we propose two new window management techniques to facilitate these tasks in the particular case of partially overlapping windows. We describe an experiment comparing four commonly used copy-and-paste techniques under four window management conditions – non-overlapping windows, partially overlapping windows, and partially overlapping ones with one of our two window management techniques. Results show that our new window management techniques significantly reduce task completion time for all copy-and-paste techniques. They also show that X Window copy-and-paste is faster than the other three techniques under all four window management conditions.

Author Keywords
Copy-and-paste, Window Management, Overlapping Windows.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: Interaction styles, Windowing systems.

INTRODUCTION
Copy-and-paste (or copy-paste) is the basic mechanism for replicating part of a document in the same or another document. Already available in early systems such as Sketchpad [28] or NLS [11], copy-paste is one of the fundamental services provided by modern graphical user interfaces. Copy-paste requires the user to specify two things: the object(s) to copy and the destination. These can be done in different orders and using various means such as the keyboard, mouse-based direct manipulation or menus. Over the years, several "standard" techniques have emerged, such as the use of Ctrl-C to copy previously-selected objects and Ctrl-V to paste them. But although these techniques are used by millions of people several times a day, the interaction is still poorly understood. The techniques are implemented differently across operating systems and among applications[1] but most importantly, to our knowledge, they have never been formally evaluated.

Copy-paste operations between two different windows usually require users to perform additional window management tasks. If the source and destination windows overlap, for example, the user often has to temporarily change the stacking order to specify the objects to copy and the destination. Yet again, the interactions and potential interferences between copy-paste and window management operations have received very little attention. A notable exception is Dragicevic's work on the Fold n' Drop technique [9] that could be applied to the particular case of drag-and-drop copy-paste.

In this paper, we propose two new window management techniques, *restack* and *roll*, to facilitate copy-paste between partially overlapping windows. We describe an experiment comparing four commonly used copy-paste techniques (keyboard shortcuts, a context menu, drag-and-drop and a technique specific to X Window) under four window management conditions: non-overlapping windows, partially overlapping windows, and partially overlapping ones with one of our two new window-management techniques. Results from this experiment show that restack and roll significantly reduce the task completion time for all copy-paste techniques. They also show that the X Window technique is faster than the three others under the four window management conditions.

The paper is organized as follows. In the next three sections, we review some of the related work, describe some common copy-paste techniques and report on a series of interviews that illustrate how they are used in practice. We then present our restack and roll techniques, detail the experiment that was conducted to evaluate them and compare the four copy-paste techniques. Finally, we discuss some implications of our results, propose some solutions to the problems identified in the paper and generalize our key ideas into the concept of *fine-grained window management*.

[1]On Microsoft Windows XP, for example, selecting text with the mouse in an overlapped window works with NotePad but not with WordPad.

CHI 2007, April 28 - May 3, 2007, San Jose, California, USA.

RELATED WORK

Although tiled windows may be more efficient for certain tasks [6, 14], the overlapping model is the *de facto* standard for all modern window systems and plays an essential part in the realization of the *desktop metaphor*. The overlapping approach supports both time-based and space-based multiplexing of the screen space by *switching* (between windows) and *splitting* (the screen) [16]. However, as the number of windows increases, it imposes time-consuming and potentially complex management tasks on the user. The goal of the work presented in this paper is to reduce this overhead. Examples of related work include techniques for leafing through stacked windows, peeling them back or making them selectively transparent to access windows underneath [3, 9, 13], and dynamic space management algorithms to reduce overlapping [4].

Users with large displays tend to leave more applications running and associated windows open [21]. Like Hutchings and Stasko [12], we believe that overlapping windows will not disappear with the advent of larger displays. First, a variety of devices will keep using small or medium-size screens. More fundamentally, although large displays make it easier to develop tiling strategies, interactions across large screen distances may become more complex and time-consuming than keeping windows together on a smaller space. Large displays are probably better used to differentiate primary and peripheral activities, i.e. for tiling tasks, not windows [21]. We anticipate that larger displays will lead to fewer maximized (full-screen) windows that completely hide others. In some cases, the previously-obscured windows may become tiled on a larger display, but in many others, they will partially overlap. Therefore, copying and pasting between partially overlapping windows will remain important.

Modifying an existing document or combining pieces from several is always easier than creating a new one. Designers of the Xerox Star said it elevated the concept of copying to the higher level of "a paradigm for creating" [25]. We indeed believe that the combined use of overlapping windows and copy-paste supports innovation and creativity [24]. Copy-paste has been studied in specific domains such as programming environments [31, 15], graphical editors [8] or ubiquitous computing environments [19, 20]. Much previous research has tried to make it "smarter" by analyzing the selected data. Citrine [27], for example, can recognize that structured text has been copied, and paste it in multiple form fields in a single operation. Other systems have been proposed to support fast copy-paste of multiple selections or text entities like phone numbers [18, 5]. In this work, we are not interested in the objects being copied, or in optimizing copy-paste for a particular domain. Rather we are interested in the low-level interactions between copy-paste and window management operations.

COPY-PASTE TECHNIQUES

In this section, we describe what we believe are currently the four most common copy-paste techniques. Note that although we focus on copy-paste, most of these techniques can also be used for cut-and-paste operations, the main difference being that the selection is deleted after the copy has been made. Other differences between copy and cut will be further explained, as needed.

Copy-paste usually starts by using the mouse to select one or more object using one or more click(s) or a press-drag-release gesture[2]. This selection might be assisted, for example by automatically snapping to the edges of objects for example. The user must then (1) activate the copy command, (2) specify the destination – in the same window or another one – and (3) activate the paste command. We will now describe several ways of accomplishing these three operations.

Using the Keyboard

Sketchpad and the Xerox Star had specific Delete, Copy and Move keys that could be used in conjunction with the pointing device. Pressing the Copy key on a Star, for example, attached the selection to the cursor, and then a mouse click specified the destination. Modern systems do not have specific keys for these functions but support keyboard-based copy-paste in a less modal way: (1) a first shortcut, e.g. Ctrl-C, causes the selection to be copied; (2) the user navigates to the destination using the mouse and/or the keyboard; (3) a second shortcut, e.g. Ctrl-V, performs the paste.

We refer to the use of keyboard shortcuts to activate the copy-paste commands as KEY *copy-paste*.

Using Menus

In addition to being accessible through keyboard shortcuts, copy-paste commands are usually found in the standard menu bar of applications, e.g., under the *Edit* item, as icons in palettes and toolbars, and in context menus accessible from the selected objects, e.g., using a right click.

Menu bars are very similar to context menus but impose additional mouse travel to reach them after selecting objects and after indicating the insertion point. Copy and paste icons in toolbars or palettes have the same problem, so we decided to focus on the use of context menus to activate the copy and paste commands.

We refer to the use of context menus to copy-paste as MENU *copy-paste*.

Using Drag-and-Drop

Drag-and-drop offers a more direct way of performing a copy-paste operation. The user simply has to press a mouse button on one of the selected objects, drag the mouse pointer to the destination and release the button. However, this technique has several problems. First, its semantics are not always easy to determine: although one can reasonably assume that dropping something on a trash icon deletes it, dropping it somewhere else might copy it or move it. As a consequence, application designers often disagree with users on what the drag-and-drop operation should do [10].

[2] As a notable exception, users of the Xerox MESA programming environment had to first specify the destination and then the text to be copied [29]. Note that objects might also be selected using the keyboard, but this does not affect the descriptions that follow.

A second problem is that the drag-and-drop requires continuous pressure on the mouse button. Besides being fatiguing and error-prone, this can make it difficult to navigate between windows to reach the destination. While keyboard shortcuts may make it possible to switch between and close windows, other functions such as minimizing, opening or moving them may be difficult if not impossible. Some applications support initiating a drag in an inactive window without bringing it to the foreground, which makes it easier to arrange the source and destination windows before the drag-and-drop operation. Another interesting solution is the use of time-based navigation techniques. As an example, the "spring-loaded" windows of the Mac OS X Finder automatically move to the foreground during a drag-and-drop if the pointer stays more than a certain time over them, go back to their original place if the pointer leaves them and stay on top if the object is dropped.

A third problem occurs when users make a too-large text selection and try to correct it [24]. In this case, pressing the mouse button inside the selected text initiates the drag-and-drop instead of initiating a new selection process. We refer to this problem as the *drag vs. subselection problem*. Note that an easy workaround is to perform a simple click to cancel the selection and then press-drag-release to make a new one.

We refer to the use of drag-and-drop to copy-paste as DND *copy-paste*.

The X Window Case

The X Window system features a simple copy-paste technique: a click on a window with the middle mouse button[3] pastes the last selection at the insertion point of that window (applications may decide to move the insertion point under the mouse pointer before pasting). This technique minimizes the number of user actions: it works as if the copy command was implicitly executed after each selection, and the mouse action that pastes also specifies the destination. In addition to this mouse-controlled *primary selection*, X also features an explicit clipboard usually accessible through the standard keyboard shortcuts we already described (i.e. Ctrl-C, Ctrl-V). Both mechanisms can be used at the same time.

One drawback of the implicit copy approach is the *volatility* of the primary X selection, as illustrated by the following scenario:

> The user selects a URL to paste in the location field of a Web browser. The field holds a previous URL. The user decides to clear it before pasting the new one: he triple-clicks on it, which selects it, and presses the Delete key. When he presses the middle mouse button, the URL he just deleted reappears...

We refer to the use of the X primary selection to copy-paste as X *copy-paste*.

[3]The X protocol was originally designed for mice with up to three buttons: left, right and middle [23]. The middle one was sometimes simulated by pressing the two others simultaneously or pressing one of them with a modifier key. Most mice now have three or more buttons, a clickable scroll wheel being often used as the middle one.

COPY-AND-PASTE PRACTICES

We interviewed twenty-one people on their copy-and-paste and cut-and-paste habits, specifically of text. We consider these people as "expert users", most of them being computer science students or engineers. Among them, ten use the X Window system, eight use Microsoft Windows and three use Apple Mac OS X.

Before asking specific questions on copy-paste, we questioned the participants on how they arrange their windows. The use of partially overlapped windows was quite common. Eleven said they either use maximized windows or partially overlapped ones, depending on the applications they run and their tasks. Four said they primarily use maximized windows, and four that they primarily use partially overlapped windows. Only two said they carefully arrange their windows by resizing and moving them following a tiling approach. These two participants also use maximized windows.

Three participants said they rarely use copy-paste. All the others said they use it very often between windows. OS X and Windows users mostly use KEY copy-paste. X Window users mostly use KEY and X copy-paste, two having said they only use X copy-paste (for text). One said he uses X copy-paste only in terminal applications where Ctrl-C is used to interrupt programs and the replacement shortcut requires both hands. This participant was a long time Windows user who switched to X Window two years ago. Only two participants said they use MENU copy-paste more than KEY copy-paste, both being Windows users and one of them having said he rarely uses copy-paste. Most people said they use MENU copy-paste. Three explained that it seemed safer (i.e. less error-prone) than the other techniques.

Three participants said they use DND cut-and-paste for text from time to time, but two said they only use it in a single window. The other participants were unable to say if a drag-and-drop moves or copies text. The *drag vs. subselection problem* was never mentioned. But when explained to the participants, thirteen said they had run into it.

Among the ten users of the X copy-paste, five mentioned the volatility problem described in the previous section. One participant said he often loses selections as he likes to select text to highlight it as he reads it. Another said he uses X copy-paste only when both source and destination windows are visible because he fears to lose the selection when he performs "complex" window management tasks like virtual desktop switching. The five X copy-paste users that didn't mention the volatility problem said they ran into it after we described it. Some said this was one of the reasons why they also use KEY copy-paste and not only X copy-paste.

We asked a few specific questions about *clipboard history tools* – tools that keep track of copy operations and support easy reuse of previously copied items. Nine participants said they had tried such tools, but do not use them anymore (six tried with Microsoft Word, three tried the KDE Klipper). We will come back to this topic in the DISCUSSION section.

All participants said they are generally happy with the way copy-paste works on their system. Some of them complained about the fact that the selection snaps to words. Others complained about the fact that they sometimes get unexpected results because of silent data conversion or formatting between the source and destination applications (e.g. between a Web browser and a word processor).

RESTACK AND ROLL COPY-AND-PASTE

Partial overlapping allows one to work on a document while keeping parts of related windows at hand (Figure 1, top). However, as illustrated by the following scenario, it quickly introduces potentially complex window management tasks when combined with even the simplest copy-paste operation:

> Héloïse is editing a text document in a window that partially covers a Web browser. She wants to copy part of the text visible in the browser into her document. She selects the relevant text in the browser with a press-drag-release gesture. As she presses the mouse button, the browser is brought to the foreground. When she releases it, the browser stays on top, partially covering her document. Héloïse presses Ctrl-C to copy the selection. She clicks on her document to bring it to the foreground and presses Ctrl-V to paste the text.

Looking at this example, one might think that the window management overhead is small, consisting in a single click on the document to bring it back to the front. But reaching this document might be difficult, since it is now behind the browser. It might even be impossible if it is fully covered (in that case Héloïse would probably use Alt-Tab to reach for it). Clicking on the document might move the insertion point and require further navigation inside it. Clicking on the window decorations solves this problem, but they are small and thus difficult to select (they may also contain dangerous controls such as the window-close button). Finally, the overall copy-paste operation changes the stacking order of the browser which, as the window second from the top of the window stack, might now cover other windows.

The root cause of these problems is that as soon as Héloïse starts interacting with a window, the system assumes it to be her primary interest. Current windowing systems provide little support to indicate secondary interest. The "focus follows mouse" policy implemented by some systems, as opposed to "click to focus", is a good example of what can be done, but it is limited to keyboard interaction. In order to further reduce this problem, we propose the following design principle:

> A left button click in a secondary window should make it of primary interest, i.e. make it active by raising it and giving it the keyboard focus. Any other interaction with a secondary window should be treated as a temporary interest indication and should be handled in a specific, appropriate way.

We propose two new window management techniques based on this design principle in order to facilitate copy-paste operations. Our previous scenario can be used to illustrate them. The first technique, *restack*, operates as follows:

> When Héloïse presses the mouse button to initiate the text selection in the browser, it is brought to the foreground. As she keeps the button pressed and starts dragging the mouse, the system infers that she is not indicating primary interest. As a consequence, when she releases the button, the browser returns to its original place in the stacking order, behind the document. It keeps the keyboard focus though, so that Héloïse can drag the selection but also use Ctrl-C to copy it. Had she decided to use a context menu, it would have been displayed in the foreground, but the browser would have stayed in its place. She can now easily drag-and-drop the selection or paste it using Ctrl-V or a context menu in the document.

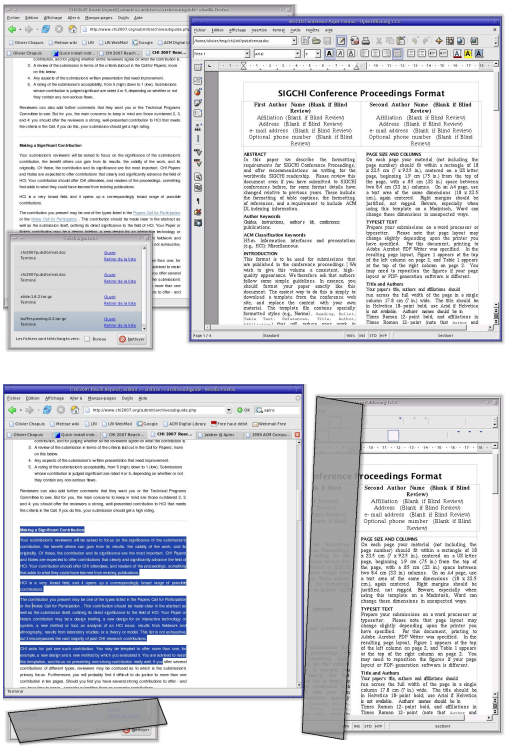


Figure 1. Rolling windows to reveal an overlapped one.

The second technique we propose, *roll*, uses a variant of the folding operation described in [3, 9] instead of automatic restacking:

> When the system infers that Héloïse is not indicating primary interest in the browser, all the windows that cover it are rolled back with a fast animation so as to fully reveal it (Figure 1, bottom). When, Héloïse finishes her selection, the windows roll back to their original state, again with an animation.

The rolling metaphor was chosen over the folding one because it hides less window content (Figure 2) and leads to less obtrusive animations. We believe that an advantage of using the restack and roll techniques for copy-paste is that

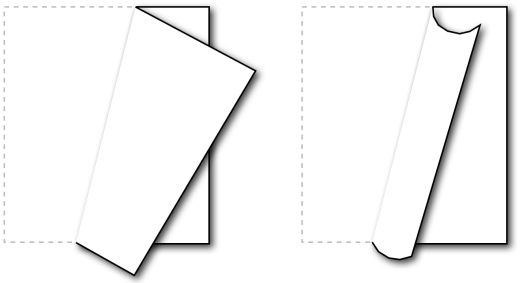


Figure 2. Folding (left) vs. rolling (right).

the window of primary interest is covered by auxiliary windows for a minimum time (during the selection), which may help users stay focused on their primary task. A drawback of these techniques is that users who want to switch focus and immediately make a selection must first click on the window to change its status. However, this limitation seems acceptable as it is quite similar to the strict "click to focus" implemented by many systems and applications.

Our design principle is based on the idea that augmenting the window management complexity will probably lead to more powerful user interfaces. One may argue that this additional complexity should be provided for expert users only.

EXPERIMENT

We conducted an experiment to compare completion times and user preferences for the KEY, MENU, DND and X copy-paste techniques between two windows under four window management conditions. We first distinguish the non-overlapping (NONOVERLAPPING) and the overlapping cases. In the overlapping case, we further distinguish three cases corresponding to the window management techniques available: the usual set of techniques (OVERLAPPING), and the restack (RESTACK) and roll (ROLL) techniques described in the previous section.

We decided to use what we thought were the most efficient variants of the MENU, DND and X techniques. Our implementation of the X technique pastes the selection under the mouse pointer (there is no notion of insertion point in the experiment). Similarly, the MENU technique pastes where the right mouse button was clicked to open the context menu. In the case of the DND technique, the initiation of a drag on a window in the background does not raise it but windows are immediately raised when the dragged object enters them in the OVERLAPPING condition. In this condition, when the drag-and-drop is not used, the only other way to raise a window is to click on it. This was decided to simplify the experiment and seemed reasonable since the two windows overlap only partially and are quite big.

Hypothesis

X copy-paste is the technique that requires the least elementary operations (free mouse movements, mouse drags, button clicks and key presses). KEY copy-paste requires two additional key presses (e.g. Ctrl-C and Ctrl-V) and DND copy-paste requires some slower mouse dragging [17] in addition to free mouse movement. These elements lead us to our first hypothesis:

H1: X *copy-paste is faster than* KEY *and* DND *copy-paste.*

This hypothesis is not so obvious in the X vs. KEY case as pressing Ctrl-C can be done while moving the mouse and X paste requires a middle button press which can be delicate to perform (e.g. in the case of a mouse wheel button). We see no obvious reason to separate KEY and DND copy-paste, and MENU copy-paste requires a lot of elementary operations (two right clicks and some menu navigation). So we proposed a second hypothesis:

H2: KEY *and* DND *(and* X*) copy-paste are faster than* MENU *copy-paste.*

Concerning the window management conditions, since the RESTACK and ROLL techniques do not require the user to raise the destination window, we proposed a third hypothesis:

H3: RESTACK *and* ROLL *techniques are faster than the* OVERLAPPING *technique for all the copy-paste techniques.*

As NONOVERLAPPING doesn't require the user to raise the destination window either, one can reasonably assume that it should be faster than OVERLAPPING. However, we see no evidence to separate (RESTACK,ROLL) and NONOVERLAPPING as RESTACK and ROLL lead to less mouse travel, but NONOVERLAPPING makes the source and destination points always visible. Finally, it is not clear whether the animations accompanying the ROLL technique are better or worse than the immediate restacking of the RESTACK technique in terms of completion time and user preferences.

Experimental Design

A repeated measures within-subject 4×4 factorial design was used. The two main factors are the copy-paste techniques (KEY, MENU, DND, X) and the window management conditions (OVERLAPPING, RESTACK, ROLL, NONOVERLAPPING). The main measure is the completion time to perform a copy-paste between two windows (Figure 3). The experiment was conducted with 18 volunteers and unpaid Computer Science students and engineers (16 males and 2 females): nine X Window users, six Windows users and three Mac OS X users. All but one had also participated in the interviews on copy-paste practices.

The experiment consists of 16 trial groups, each group consisting itself in a series of at least 4 trials. Within each group, the copy-paste technique and the window management condition are fixed. Copy-paste techniques are not intermixed. As an example, the subject performs four trial groups of X copy-paste with NONOVERLAPPING, OVERLAPPING, RESTACK and ROLL; then four groups of DND copy-paste with ROLL, RESTACK, OVERLAPPING and NONOVERLAPPING; etc. Orders of the techniques and of the window management conditions

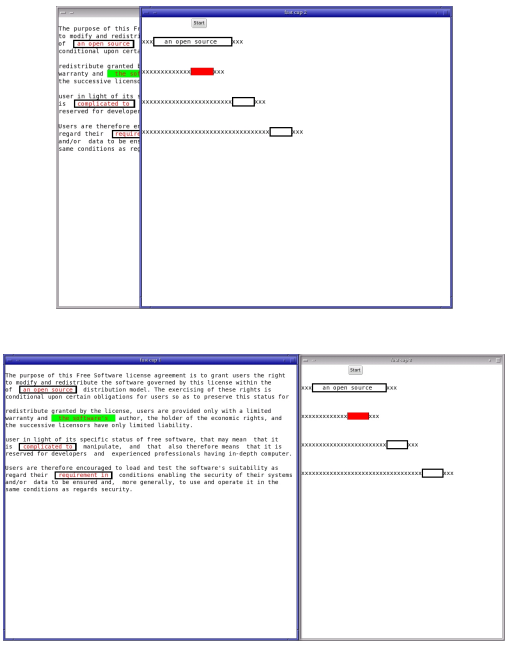


Figure 3. Overlapping and non overlapping conditions. Images show the second copy-paste of a trial with current text source and destination highlighted. Other sources and destinations have been framed for the convenience of the reader.

were pseudo-randomly balanced across participants following a Latin-square design.

A trial consists of a series of four copy-paste actions. The subject first presses a "start" button at the top of the right window. The first text to copy appears highlighted in green in the left window and the corresponding paste area in red in the right one. The subject selects the text and executes the copy-paste operation. As soon as the text is pasted, the next copy-paste areas are highlighted in the two windows, below the ones that were just used (see Figure 3). When the fourth copy-paste is done, the chronometer is stopped. The number of successful operations for the current trial is presented to the subject with some indication of his progression in the experiment. The subject can take a short break and then move on to the next trial by clicking the "start" button again. Note that we do not consider these repeated copy-pastes as a natural task. This is simply a way of obtaining as many copy-paste completion times as possible in a minimum amount of time. These times should be representative, similarly to the classical Fitts' pointing experiments that use back and forth pointing.

The four texts to copy during the trials have the same number of characters and the same length (a monospace font is used). Their position and the position of their corresponding paste area are fixed. These positions were chosen so as not to be favorable to a particular window management condition (the first and second copy-paste are favorable to RESTACK and ROLL, the third and fourth ones to NONOVERLAPPING and OVERLAPPING). In the overlapping case, two are fully visible while the two others are initially half-covered by the right window (Figure 3, top). In the NONOVERLAPPING condition, the right window is just moved further to the right to suppress overlapping and resized to keep it fully on screen.

Each trial group starts with a training period used to explain the technique to the subject. Subjects are allowed to train as long as they want ("train until you feel at ease with the technique"). The first trial actually starts when the subject presses a button. Subjects are instructed to "perform as fast as possible without errors". The group finishes when the subject has successfully performed four copy-paste of each of the four texts (i.e. at least four trials and sixteen successful copy-paste). Pasting can be done anywhere in the paste area, which is made of several spaces. The selection mechanism is also space-tolerant. The error policy is otherwise strict: the subject must perform the perfect interaction.

Apparatus

The restack and roll window management techniques were implemented inside the Metisse window system [7]. The software used for the copy-paste experiment was written in C using the GTK+ toolkit. The experiment ran on a 2.66 GHz bi-processor PC running Linux with a powerful graphics card connected to a 1280x1024 LCD display. The mouse used was a standard optical one with two buttons and a clickable wheel that could be used as a third (middle) button. The default linear X Window mouse acceleration was used. Instructions and source code needed to reproduce the experiment are available from `http://insitu.lri.fr/metisse/rock-n-roll/`.

Results

We analysed the data using a repeated measures analysis of variance (ANOVA), with completion time to perform one copy-paste (CT in milliseconds) as measure, subject as a random effect factor, copy-paste techniques and window management conditions as fixed effect factors. Our major interest is in the interactions between copy-paste techniques and window management conditions. We used JMP [22] to perform the ANOVA with the Restricted Maximum Likelihood (REML) method. Erroneous copy-paste operations were removed from our data: on a total of 5229 trials, 4608 error-free completion times were taken into account (18 participants $\times$ 4 copy-paste techniques $\times$ 4 management conditions $\times$ 4 texts $\times$ 4 repetitions).

Figure 4 shows almost all the results of this analysis. The copy-paste techniques reveal significant effects ($F_{3,4575} = 2334$, $p < 0.001$) as do the window management conditions ($F_{3,4575} = 248$, $p < 0.001$), and there is no evidence of an interaction between these factors ($F_{9,4575} = 1.01$, $p = 0.43$). As no interaction appears to be present, we focus our analysis on the main effects. We use the Tukey HSD (honestly significant difference) test with $\alpha = 0.05$. Figure 5 details the results of these tests for our two main factors.

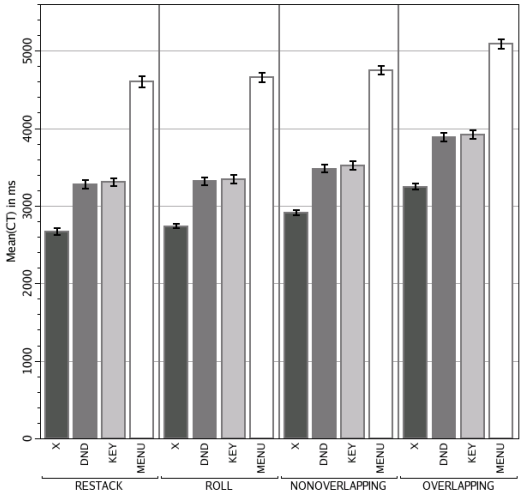


Figure 4. Completion time for each copy-paste technique, grouped by window management condition. Error bars show standard error.

	X	DND	KEY	MENU
X	0 0 0	-599 -659 -539	-633 -693 -574	-1883 -1943 -1824
DND	599 539 659	0 0 0	-34 -93 25	-1284 -1343 -1224
KEY	633 574 693	-34 -25 -93	-0 -0 -0	-1249 -1309 -1190
MENU	1883 1824 1943	1284 1224 1343	1249 1190 1309	-0 -0 -0

	RESTACK	ROLL	NONOVER.	OVER.
RESTACK	0 0 0	-50 -109 9	-200 -259 -140	-570 -629 -510
ROLL	50 -9 109	0 0 0	-150 -209 -90	-520 -579 -460
NONOVER.	200 140 259	150 90 209	-0 -0 -0	-370 -429 -310
OVER.	570 510 630	520 460 579	370 310 429	0 0 0

Figure 5. Tukey crosstab: techniques (top) and window management conditions (bottom). A cell contains the means difference and the lower and upper bound of the confidence interval. Underlined cells are significant.

First, we examine the copy-paste techniques. X copy-paste is significantly faster than KEY and DND, and KEY and DND are significantly faster than MENU (H1 and H2 are supported). We found no significant difference between KEY and DND (this is also the case for each window management condition). A practical equivalence test [30] with a threshold of 100 ms (3% of the means) gives a positive result ($p = 0.002$). However, as we explained, our implementation of the DND technique immediately raises a window in the OVERLAPPING condition when the dragged text enters it. So, one may assume that with a "spring-loaded"-like implementation, we would have found a significant difference between means under OVERLAPPING, caused by the delay. It is interesting to note that X is 18% faster than KEY which is probably the most popular technique among "expert" users.

We now examine the window management conditions. H3 is supported: ROLL and RESTACK are significantly faster than OVERLAPPING (and NONOVERLAPPING is also faster than OVERLAPPING). We found no significant difference between RESTACK and ROLL[4]. More surprisingly, ROLL and RESTACK are significantly faster than NONOVERLAPPING. The difference between means is small. However, X copy-paste is 10% faster with RESTACK than with NONOVERLAPPING. On the other hand, RESTACK is 18% faster than OVERLAPPING (for all techniques but MENU, where RESTACK gives only a 10% speed-up). Finally, we note that X with RESTACK is 32% faster than KEY in the OVERLAPPING condition. For such an elemental operation as copy-paste, this is a huge improvement. See Figure 6 for more numbers regarding the combinations of copy-paste techniques with the window management conditions.

Level								CT Mean
MENU,OVERLAPPING	A							5085
MENU,NONOVERLAPPING		B						4746
MENU,ROLL		B						4656
MENU,RESTACK		B						4600
KEY,OVERLAPPING			C					3919
DND,OVERLAPPING			C					3883
KEY,NONOVERLAPPING				D				3522
DND,NONOVERLAPPING				D	E			3475
KEY,ROLL					E	F		3342
DND,ROLL						F		3316
KEY,RESTACK						F		3305
DND,RESTACK						F		3277
X,OVERLAPPING						F		3244
X,NONOVERLAPPING							G	2906
X,ROLL							H	2737
X,RESTACK							H	2667

Figure 6. Means and significance for the combination of the copy-paste techniques and of the window management conditions. Levels not connected by the same letter (A,B,C, ...) are significantly different.

The overall error in the experiment was 5.58% (Figure 7). However, a new ANOVA (similar to our main ANOVA, but with errors as the measure) only shows that XS and DND are less error-prone than MENU: the technique is a significant factor, but the window management condition is not and there is no evidence of an interaction between the two factors.

In the OVERLAPPING condition, as we explained, two of the texts to select in the left window are half covered by the right one, while the two others are not. Since the lengths of the texts are equal, this allows us to compare the selection time

[4]However, with the less robust Student's t test we found a significant difference ($t = 2.16$, $p = 0.03$). The difference between the means is of 50 ms which can be compared to the roll back animation which takes 150 ms.

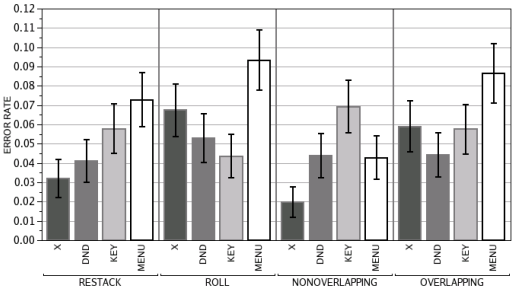


Figure 7. Error rate for each copy-paste technique, grouped by window management condition. Error bars show standard error.

for overlapped vs. non overlapped text. We performed an ANOVA similar to our main ANOVA, but we removed the NONOVERLAPPING data, added "text overlap" as a new fixed effect factor and took selection time as the measure. Text overlap has a strong effect. Subjects performed the text selection faster when the text was not overlapped (with a significant difference between means of 93 ms for a mean selection time of 858 ms). The only other effect is a small interaction between the text overlap condition and the window management conditions. The difference between means is more important with ROLL (129 ms) because the selection time is faster with ROLL for non overlapped text (non significant on its own) and slower for overlapped text (significant vs. OVERLAPPING, but not RESTACK).

KEY copy-paste is bi-manual: the left hand can be used to activate the keyboard shortcuts while the right hand controls the mouse. We tried to see how far the subjects did use synchronised bi-manual interaction. The average time between the moment subjects place the cursor in the paste area and the moment they press Ctrl-V is of 454 ms in average. This is big, but goes down to a value closer to 200 ms for some subjects. The average time between the moment subjects finish the selection and the moment they press Ctrl-C is 368 ms. During this time, some subjects move the mouse: we measured an average distance of 97 pixels in the non overlapping case (for some subjects this value is close to 0, but for some others, it is closer to 300). Subjects also move the mouse between the Ctrl-C press and release: we measured an average distance of 269 pixels in the non overlapping case (with no huge difference between subjects).

To estimate the time lost by pressing Ctrl-C, we performed an ANOVA similar to our main ANOVA but with only the X and KEY techniques (and all the window management conditions) and where the measure is the time between the end of the text selection and the moment where the subject presses the middle mouse button (in the X case) or positions the text cursor with a left click (in the KEY case) in the paste area. The copy-paste techniques and the window management conditions are significant factors and there is no evidence for an interaction between the factors. We get a significant difference between means for the X technique vs. the KEY technique of 223 ms ($p < .001$).

Qualitative Results

At the end of the experiment, participants were asked to rate the copy-paste techniques and window manager conditions.

Of the eighteen subjects, fifteen said they preferred X copy-paste. Two said they preferred DND (both liked the feedback given by this technique). Only one said he preferred KEY. Thirteen subjects cited MENU as the worst technique, two cited DND and three cited KEY. KEY was cited ten times as the second preferred technique, DND was cited nine times at this place and MENU three times (with some ex aequo). X is clearly the technique that the subjects preferred. KEY and DND are the second preferred techniques and MENU is clearly disliked. It is interesting to note that between the nine Windows or OS X users, seven said they would like to have the X technique available on their system.

Eight subjects preferred to perform copy-paste under the NONOVERLAPPING condition (eight subjects cited it in second place). Seven subjects cited RESTACK as their preferred way to run the experiment (eight subjects cited it in second place). Most subjects said that they were disturbed by the animation of the ROLL techniques, three subjects placed it first (they liked the animation and the graphics), three subjects at the second position and ten at the third position. All the subjects preferred RESTACK and NONOVERLAPPING to OVERLAPPING. Only two subjects did not cite OVERLAPPING as the worst technique. Both found the animation produced by ROLL strongly disturbing. RESTACK was thus accepted: because of this technique a reasonable number of subjects preferred an overlapping context to a non overlapping one, and most of the others placed it second. Moreover, two subjects asked if RESTACK could be made available on their system.

ROLL was not similarly appreciated, which is unfortunate because it gives more feedback about what is going on than RESTACK (only two subjects placed ROLL before RESTACK, and one placed both techniques second). One possible reason for this is that the subjects were told to perform the experiment as fast as possible, which the 150 ms animation didn't help (especially in our repeated task). Moreover, in the case of a MENU or DND copy-paste on an overlapped text, the unrolling of the right window over the text can make it difficult to open the context menu or drag it. Two subjects mentioned this as a problem. However, a third one claimed it helped him move to the left to grab the text.

The subjects easily answered our questions regarding their preferred copy-paste techniques and their preferred window management conditions. We also asked whether they had specially liked or disliked any combinations of them. We got very few answers. Two subjects who preferred the X technique said they preferred the DND technique under the NONOVERLAPPING condition. Three subjects made some remarks regarding the interaction between DND/MENU and ROLL (see the previous paragraph). A few subjects who said they disliked MENU added that they particularly disliked it in the OVERLAPPING condition. Two subjects who already preferred the X technique, said it was really better in the OVERLAPPING condition.

DISCUSSION

X copy-paste is fast, simple and can be used in conjunction with KEY and MENU copy-paste. Most people who tried it like it. X Window has unfortunately no equivalent technique for cut-and-paste. One could probably implement a drag-select-and-cut command, map it to a specific mouse button and reuse the middle button for pasting. However, as the interviews confirmed, the volatility of the selection clearly poses some problems. Selection history tools could certainly help, but none of the interviewed people were in the habit of using them. We believe the main problem with these tools is that they are usually accessible from a system or application menu bar, but not directly from the place where the user wants to paste. In the case of X copy-paste, we suggest that a long middle button press should pop up a context menu presenting the selection history. This idea can also be applied to KEY copy-paste: pressing Ctrl-V and holding the V key pressed could also pop up the history. The user could then circulate in it by repeatedly pressing the V key.

Restack and roll are currently used on a daily basis by the first author of this paper and a student. Both use the techniques several times a day and the first author even developed some placement strategies to take advantage of them. Most text documents being left-aligned (sometimes justified), overlapping them on the right side usually leaves more content visible than on the left side. As a consequence, when writing a paper, the author usually displays auxiliary documents (e.g. other papers, Web pages) on the left side of the screen, partially covered by a window on the right side showing the paper. The student often uses the restack technique to paste command line templates found on Web pages in a terminal that overlaps the browser and is sometimes fully surrounded by it. One interesting point is that both users diverted the techniques. As an example, they often make arbitrary selections just to temporarily expose an overlapped window. This can be viewed as the counterpart of the folding operation described in [3] which was designed to temporarily look behind a single window by grabbing one of its corners and peeling it back.

The idea of differentiating user interactions with the primary window from those with secondary windows opens an interesting design space. With restack and roll, we proposed specific actions that temporarily expose a window when the user selects some of its content. One might ask what should happen when the user interacts in other ways with a secondary window. What should happen, for example, when the user starts dragging the scrollbar of a partially covered window? One possibility would be to uncover it when the drag starts (e.g. using restack or roll), let the user manipulate the scrollbar and either put it back to its original place if the button is released outside the window or make it of primary interest otherwise.

Similar questions could be asked for other interactions, other types of widgets, unused window space or even the desktop. Selecting an icon on the desktop, for example, could temporarily expose its surroundings by rolling nearby windows, or rendering them using selective transparency [13] or multiblending [2]. In an attempt to generalize these ideas, based on the design principles we previously described and similar arguments given in [1, 9, 13], we propose the concept of *fine-grained window management*, defined with the following goal:

> Take into account the context of user actions on windows (e.g. the window type or content, or its surroundings) to execute the most appropriate window management command.

Creating a fine-grained window manager poses a number of technical problems. It should, for example, have some knowledge about the relations between windows and their internal structure, e.g., the widgets they contain and the potential user actions on them. In order to work with a wide range of applications, our implementation of the restack and roll techniques relies on several "hacks" to monitor mouse activity and the various X selection mechanisms. Accessibility APIs provide clean ways to figure out internal window structures that can help implement fine-grained window management techniques [26]. But more than this, we believe there is a need for new, richer, bi-directional communication protocols between the window manager and the applications.

CONCLUSION

In this paper, we examined the problems related to copy-paste between partially overlapping windows. We proposed two new window management techniques, *restack* and *roll*, to solve some of these problems. We described an experiment that was conducted to evaluate these techniques and compare four common copy-paste techniques. Results show that X Window copy-paste is faster than the use of keyboard shortcuts, context menus or drag-and-drop. They also show that restack and roll significantly improve the four copy-paste techniques. Restack and roll were designed according to the idea that user interactions with windows of primary interest could differ from those with secondary windows. We intend to continue exploring this idea to develop the more general concept of fine-grained window management that takes the context of user actions into account.

ACKNOWLEDGEMENTS

We would like to thank the participants of the interview and of the experiment for there time and cooperation. We also thank J.D. Fekete, P. Dragicevic, J.B. Labrune, H. Goodell, A. Cockburn, M. Beaudouin-Lafon, W. Mackay, C. Appert, E. Pietriga and Y. Guiard for helpful discussions about this work. Finally, we thank the anonymous reviewers for their useful comments and suggestions about this paper.

This work has been partially funded by the French *ACI Masses de données* (Micromégas project).

REFERENCES

1. C. Appert, M. Beaudouin-Lafon, and W. Mackay. Context Matters: Evaluating Interaction Techniques with the CIS Model. In *Proceedings of HCI'04*, pages 279–295. Springer Verlag, 2004.
2. P. Baudisch and C. Gutwin. Multiblending: displaying overlapping windows simultaneously without the draw-

backs of alpha blending. In *Proceedings of CHI '04*, pages 367–374. ACM Press, 2004.

3. M. Beaudouin-Lafon. Novel interaction techniques for overlapping windows. In *Proceedings of UIST '01*, pages 153–154. ACM Press, 2001.

4. B. Bell and S. Feiner. Dynamic space management for user interfaces. In *Proceedings of UIST'00*, pages 239–248. ACM Press, 2000.

5. E. A. Bier, E. W. Ishak, and E. Chi. Entity quick click: rapid text copying based on automatic entity extraction. In *Extended abstracts of CHI '06*, pages 562–567. ACM Press, 2006.

6. S. Bly and J. Rosenberg. A comparison of tiled and overlapping windows. In *Proceedings of CHI '86*, pages 101–106. ACM Press, 1986.

7. O. Chapuis and N. Roussel. Metisse is not a 3D desktop! In *Proceedings of UIST '05*, pages 13–22. ACM Press, 2005.

8. W. Citrin, D. Broodsky, and J. McWhirter. Style-based cut-and-paste in graphical editors. In *Proceedings of AVI '94*, pages 105–112. ACM Press, 1994.

9. P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proceedings of UIST '04*, pages 193–196. ACM Press, 2004.

10. E. Dykstra-Erickson and D. Curbow. The role of user studies in the design of OpenDoc. In *Proceedings of DIS '97*, pages 111–120. ACM Press, 1997.

11. D. C. Engelbart. Authorship provisions in augment. In *Proceedings of COMPCON '84*, pages 465–472. IEEE, 1984.

12. D. Hutchings and J. Stasko. Revisiting Display Space Management: Understanding Current Practice to Inform Next-generation Design. In *Proceedings of GI '04*, pages 127–134. Canadian Human-Computer Communications Society, 2004.

13. E. W. Ishak and S. K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *Proceedings of UIST '04*, pages 189–192. ACM Press, 2004.

14. E. Kandogan and B. Shneiderman. Elastic Windows: evaluation of multi-window operations. In *Proceedings of CHI '97*, pages 250–257. ACM Press, 1997.

15. M. Kim, L. Bergman, T. Lau, and D. Notkin. An ethnographic study of copy and paste programming practices in OOPL. In *Proceedings of ISESE '04*, pages 83–92. IEEE, 2004.

16. B. Lampson. Personal distributed computing: the alto and ethernet software. In *Proceedings of the ACM Conference on The history of personal workstations*, pages 101–131. ACM Press, 1986.

17. I. S. MacKenzie, A. Sellen, and W. A. S. Buxton. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of CHI '91*, pages 161–166. ACM Press, 1991.

18. R. C. Miller and B. A. Myers. Multiple selections in smart text editing. In *Proceedings of IUI '02*, pages 103–110. ACM Press, 2002.

19. B. A. Myers. Using handhelds and PCs together. *Communications of the ACM*, 44(11):34–41, 2001.

20. J. Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of UIST '97*, pages 31–39. ACM Press, 1997.

21. G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. Hutchings, B. Meyers, D. Robbins, and G. Smith. Scalable Fabric: Flexible Task Management. In *Proceedings of AVI '04*, pages 85–89, 2004.

22. SAS Institute Inc. *JMP Version 6*. SAS Institute Inc., Cary, NC, 1989-2005.

23. R. Scheifler and J. Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, 1986.

24. B. Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction*, 7(1):114–138, 2000.

25. D. Smith, C. Irby, R. Kimball, W. Verplank, and E. Harslem. Designing the Star user interface. *Byte*, 7(4):242–282, 1982.

26. W. Stuerzlinger, O. Chapuis, D. Phillips, and N. Roussel. User Interface Façades: Towards Fully Adaptable User Interfaces. In *Proceedings of UIST'06*, pages 309–318. ACM Press, 2006.

27. J. Stylos, B. A. Myers, and A. Faulring. Citrine: providing intelligent copy-and-paste. In *Proceedings of UIST '04*, pages 185–188. ACM Press, 2004.

28. I. Sutherland. *Sketchpad, a man-machine graphical communication system*. PhD thesis, Massachusetts Institute of Technology (USA), 1963.

29. R. Sweet. The MESA programming environment. In *Proceedings of the ACM SIGPLAN 85 symposium on Language issues in programming environments*, pages 216–229. ACM Press, 1985.

30. W. W. Tryon. Evaluating statistical difference, equivalence, and indeterminacy using inferential confidence intervals: An integrated alternative method of conducting null hypothesis statistical tests. *Psychological Methods*, 6:371–386, 2001.

31. G. Wallace, B. Robert, and E. Tempero. Smarter cut-and-paste for programming text editors. In *Proceedings of AUIC '01*, pages 56–63. IEEE, 2001.

PageLinker: Integrating Contextual Bookmarks within a Browser

Aurélien Tabard[1,2] *tabard@lri.fr*
Wendy Mackay[1,2] *mackay@lri.fr*
Nicolas Roussel[2,1] *roussel@lri.fr*
Catherine Letondal[3] *letondal@pasteur.fr*

[1]INRIA F-91405 Orsay, France
[2]LRI - Univ. Paris-Sud & CNRS F-91405, Orsay, France
[3]Institut Pasteur F-75015, Paris, France

ABSTRACT
PageLinker is a browser extension that contextualises navigation by linking web pages together and allows navigation through a network of related web pages without prior planning. The design is based on extensive interviews with biologists, which highlighted their difficulties finding previously visited web pages. They found current browser tools inadequate, resulting in poorly organised bookmarks and rarely used history lists. In a four-week controlled field experiment, PageLinker significantly reduced time, page loads and mouse clicks. By presenting links in context, PageLinker facilitates web page revisitation, is less prone to bookmark overload and is highly robust to change.

Author Keywords
Biologists, Bookmarks, Browsers, Contextual bookmarks, PageLinker, Participatory design, Web Navigation, WWW

ACM Classification Keywords
D.2.2 [Design Tools & Techniques]: User Interfaces. H.5.2 [User Interfaces]: Evaluation/methodology, User-centered design. H.5.4 [Hypertext/Hypermedia]: Navigation, User Issues

INTRODUCTION

The World Wide Web has expanded dramatically in the past decade, with huge increases in the number of users, web pages and complexity of content. Unfortunately, at the level of user interaction, web browsers have not kept pace. Early user aids for finding previously visited pages, e.g., bookmarks and history, have evolved little since their introduction in the early 1990's [3]. Even though revisitation accounts for half or more of visited pages [6,9,25] studies show that revisitation tools are rarely used [6,25,27].

We have been studying a particularly web-intensive group of users, research biologists, who have reorganised their work around the internet. They treat the Web as an enormous, constantly searched database and also as an analysis tool. They repeat collections of tasks, revisiting the same sets of pages over and over again, browsing sequentially and in parallel as they analyse data sets and pursue hypotheses. Unfortunately, their improvised, fluctuating workflow is often poorly supported by the websites they use. The data pages they seek may require long navigation paths through huge hierarchical directories, and are unlikely to contain direct links to the analysis programs they will apply to this data. For them, as others, bookmarks and history pages are insufficient.

How can we facilitate page revisitation tasks? Automation tools that allow users to build and play common scenarios offer one solution. However they usually require too much advance planning: Biologists must rethink their workflow at each navigation step and each decision depends upon multiple situated factors [23], including time available, knowledge of server loads or difficult-to-articulate factors such as one's intuitions about whether certain results are 'normal'. Visualisation tools that graphically illustrate previous navigation steps are another possibility. However these require a great deal of screen real estate and focus attention away from the primary navigation task.

Based on these problems, observed with current browsers and other navigation tools (visualization, automation), we developed PageLinker, a browser extension that allows users to *contextualise* their navigation by associating web pages together, i.e. to create and present links only on specific pages or set of pages.

We describe our preliminary study of biologists at several research institutions, with insights gained from interviews, observations, brainstorming sessions and workshops. We then discuss implications for the design of contextual bookmarks, including a review of the relevant literature. We next present the evolution of PageLinker and describe a controlled field experiment to evaluate it. We conclude with an analysis of the results and discuss implications for future research.

PRELIMINARY STUDY

We have been working closely with research biologists for the past eight years [15] in a variety of participatory design projects. Although not necessarily computer programmers, these biologists are highly experienced internet users who have modified their work practices to take advantage of the wealth of biological data and analysis programs available on the web. This study reported here focuses on the problems they face when navigating the web.

We selected 20 biologists who had recently used on-line biological data and analysis programs as an integral part of their research. We conducted videotaped interviews in their labs and asked them to replay, with their own data, their latest analysis using a web application. We also asked them to search for specified information in their research institution's online documentation. We used a talk-aloud protocol, asking them to explain their choices and what bothered them, as they performed these tasks. We also organised a video-brainstorming [19] workshop that focused on organising analysis workflows, either by using and possibly editing history data or by assembling analysis resources on the Web.

Illustrating the navigation problem

The following scenario illustrates a typical navigation session for a biologist studying a protein. Ann needs to explore alternative hypotheses before conducting a time-consuming lab experiment. She begins by collecting data: From the Biology department's homepage, she follows links to the protein database page. Unfortunately, it doesn't offer links to relevant analysis tools and she must browse a huge, hard-to-navigate hierarchical directory with hundreds of links spread over many pages. She eventually finds the relevant page and checks the research literature to see if similar forms of the protein appear. She then looks for the protein sequence in two different databases to find out if different DNA sequences are associated with the protein. She encounters incompatible data formats, forcing her to transform the data before using her chosen analysis protocol.

The lack of relevant links in the data pages makes it difficult for Ann to move from one step to the next. Even when she does find appropriate online resources, she has trouble keeping track of them. Several weeks later, when she decides to analyse a new set of data, she has to recreate her initial search process in order to find the same pages again. Like others in our study, Ann rarely uses bookmarks or history pages, and instead relies on Post-it™ notes, e-mail and search engines to find previously visited sites.

Observations

Several recurrent themes emerged from our interviews and the workshop on online data management:

Habits: Most biologists:

- have bookmarks but often prefer to use search engines, email and physical Post-it™ notes;
- reach previously bookmarked sites via search engines, because their bookmarks are difficult to browse;
- save temporary results or alternative data formats; and
- rarely customize web forms, even when possible.

Software strategies: Biologists are conservative software users. They prefer a stable and predictable set of tools [18] and tend to use techniques they already know rather learning a new, potentially better one. Most stay with a single Web server if it provides all the tools they need, even though better tools might exist on other servers. Most biologists are usually skeptical of pipelining and automation tools that support biological protocols, since the learning curve is often steep and the benefits are usually limited.

Interaction: Biologists' purposes and procedures change rapidly. Unlike programming, constructing a biological online protocol is not fully algorithmic and requires human judgment along the way. Biologists check the accuracy or significance of results and decide whether and how to carry out an analysis using complex criteria that would be difficult to automate. A biologist might decide to use different processes, proceed with full data or extract subsets depending upon on the characteristics of the data and her current research goal.

Equivalent objects: Data formats are often incompatible: the output of one tool may not be interpretable as input by another tool. Biologists are thus forced to edit intermediate results and end up managing collections of "equivalent" data objects, including:

- same data in different formats needed by different tools
- different versions of the same data, e.g., two versions of an annotated genome.

Data flows: Biologists create diverse data flows, piping the output of one program into another as well as reformatting, transforming, filtering and extracting data [22]. They use copy-paste to chain these steps, which is not supported by automated tools. Like Tauscher & Greenberg [25], we found that they preferred to replay a path rather than using history to access a specific page.

Related work

Our observations match findings in the research literature with respect to re-visiting web pages and recording and connecting resources over the internet. Tauscher & Greenberg define the Web as a 'recurrent system' [24] and report that 58% of pages are revisited. Weinreich et al. [27] reported 46% and Cockburn & McKenzie [9] reported 81% in their respective studies.

Unfortunately, the history and bookmarks mechanisms provided by browsers are not sufficient to support web page re-visitation [6, 25, 27]. Web browsers provide both short-term (*back* and *forward* buttons) and long-term history mechanisms (global history lists). Although *back* is used relatively frequently (14% of navigation actions), global

history lists are rarely used [6, 7, 24], only about 0.2% of all page requests [27]. Stored history information is usually very limited, capturing only the last time (and possibly the first) a URL was visited. This makes it difficult for users to find a page accessed from a well-known site at a specific date. For example, if a biologist follows a path through a series of websites to fulfill a protocol, and one of those sites is visited later, the earlier path will no longer appear in the history file.

Studies of navigation paths show that bookmarks are not a panacea for solving the problem of page access [1]. The changing nature of the web and users' changing interests [20, 25] often cause classification and relevance problems. Page titles are often obscure or too long to be displayed in a menu [20]. Bookmark lists tend to grow over time as users add new pages without removing unused ones [9], providing "neither a reminding function nor a context of relevance"[13]. If users do not constantly edit and prune their lists, they end up with inappropriate and uninteresting URLs, little better than no bookmarks at all [24].

Graphs of navigation history provide an alternative to history lists [12], situating current activity within previously used paths. However, graphs require additional screen space and force users to shift between their primary browsing tasks and a secondary location task. An interesting alternative is WebView [8], a browser enhancement that integrates several revisitation capabilities into a single display, resulting in a compact revisitation tool. While WebView is promising, it focuses mainly on providing a better interaction with the global history.

Another trend in revisitation tools is to automate navigation. However, the instability of the Web introduces problems: changes in page content, URLs, and data formats can "break" formerly correct automation sequences. Other common problems with automating complex workflows are the lack of transparency, as users search for the cause of unexpected results, and the lack of interaction, when they need to explore possible changes to a sequence. For example, one biologist commented that he "needs to redo the protocol step by step because there is no convenient way to access the problem source directly". The process of navigating through various websites acquaints biologists with changes on the server, new programs, and new layouts that might provide easier access to some pages, helping them to gather knowledge about their virtual environment. Teevan et al. [26] argue, in another context, that directed situated navigation reduces the quantity of information that users need to specify and provides the context they need to help them understand the results they obtain.

Initial design choices

Based on our interviews and insights gained from earlier studies [15, 16], we decided to focus on supporting the biologists' process of analysing web-based data. We wanted to create a tool that fit with their existing work patterns, so they could use familiar work practices and their own data and not be forced to add additional tasks. We based the design on our observation that biologists use *copy* to extract data from one web page and *paste* to enter it into an analysis form[1], thus identifying which pages make sense to link together.

We selected the Firefox web browser because it is available on Mac OS X, Linux, and Windows and was already used by half the biologists in the study. Installing a Firefox extension is easy: users need only click on the link of the extension they want to install. Firefox can also track *copy* and *paste* events, making it possible to automatically generate the links we observed above. PageLinker takes advantage of this functionality and allows users to contextualise their navigation, automatically linking web pages as the biologist cuts and pastes between them. Later versions of PageLinker also allow users to create these contextual bookmarks manually and offer feedback by showing the most recently created link in the menu.

ITERATIVE DESIGN OF PAGELINKER

Phase 1: Initial implementation

The first version of PageLinker focused on creating links invisibly, based on the user's cut, copy and paste actions. PageLinker overrides copy, cut and paste events: When a *copy* or *cut* event is detected, it records the page (title, URL, and date) and, as soon as a *paste* event is detected, creates a link between the two pages. The *copy* (or *cut*) page thus points to the page where the *paste* occurred. Our interviews and workshops indicated that biologists rarely use output data from one page when they need to fill out a new form. Instead, they usually edit the data, either to address incompatible data formats or to refine their request. We link the page of the most recent copy event to the current paste page, without considering the contents of the clipboard. We can thus accommodate the "equivalent objects" mentioned earlier, where the physical data formats are different but, from the biologist's perspective, the content is the same.

PageLinker uses XUL, JavaScript and RDF[2]. The new definitions of copy, cut and paste items from the menus are implemented with XUL, an XML-based language used to define interfaces. JavaScript handles user interface actions and manages data. We override the clipboard *shortcuts* events by grabbing Ctrl-C/X/V on Windows and Linux or Cmd-C/X/V on Mac OS. We use RDF to implement file recording of contextual bookmarks. A collection of RDF statements represents a labeled, directed graph. Figure 1 shows the graph illustrating a link between two pages. Each page is a node pointing to the pages it is related to. Since RDF allows only simple oriented graphs, our structure is redundant for bi-directional links.

---

[1] We use the term *form* to refer to pages that require the user to enter data. Some of these forms also generate data.

[2] See: http://developer.mozilla.org/

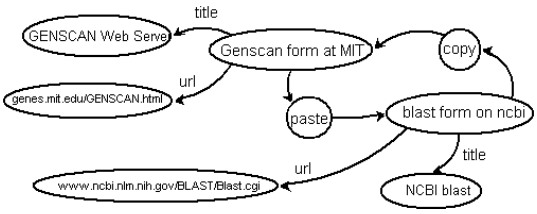


Figure 1: Graph outline of a link between the GENSCAN results (copy) and the BLAST form (paste).

Each page points to its descriptors, e.g., title and URL, as well as a *copy* node and a *paste* node. The copy node points to the list of pages where data was copied from the current page and the paste node points to the list of pages from where data was pasted into the current page. The RDF is queried through a template-based request language supported by XUL in order to map the contextual bookmarks display and the RDF file. When the RDF is modified, its corresponding UI component is automatically updated.

How do we decide which part of the URL to use? If we use the entire URL, the result is too restrictive: we get a large number of pages with only minor variations among them. If we use the root URL, i.e. the main site at the top of a hierarchy of web pages, we only get the main site and lose all of the interim searching the user has done. PageLinker uses the full URL, minus the query string. The resulting contextual bookmarks are specific to a particular web form, rather than a particular result or the whole server.

Iterative design based on user feedback

PageLinker was created using a participatory design process together with biologists at the Institut Pasteur. We tested the first version, PageLinker 0.1, with six biologists who installed it and provided constructive feedback via interviews and direct observation. We chose the simplest design possible: links were based on invisibly-captured copy-paste events and users interacted with PageLinker via the *Shortcuts* menu (Figure 2).

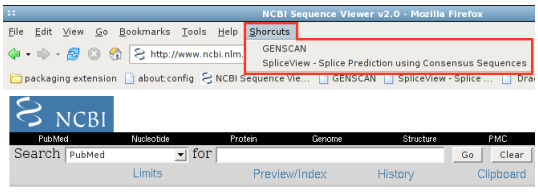


Figure 2: Shortcuts contextual menu (PageLinker 0.1)

Over time, users found that obsolete items had ended up on the Shortcuts menu and asked us to remove them. At this point, some users discovered how to use PageLinker to manually add links between pages, an example of co-adaptive behavior [17]. They used the Ctrl-C and Ctrl-V shortcuts on pages without entry forms, for example, between one page with press reviews and another with the referenced newspaper articles or between an application form and the relevant documentation page.

Although using control-keys was fine for some users, others requested a more convenient interface for manually linking of contextual bookmarks. Several people liked the concept but found it annoying to copy-paste when it was not required for the task at hand. They commented that they would decide to link back to a previous page only *after* they had successfully identified an interesting subsequent page. Using the copy-paste technique required returning to the previous page and generated meaningless extra actions.

Based on this feedback, we conducted a participatory design workshop to explore simpler ways to create links between pages. We worked together with the biologists to create video prototypes [19] that envision scenarios for linking to a desired destination from a previous page. We created prototypes of three linking strategies: via open pages or tabs, via the last visited page and via the global history.

PageLinker 0.2 implemented all three methods. We added a *link to* menu to the toolbar (Figure 3) that presents a list of all the browser's open web pages (both on tabs and in other windows) and the seven most recently visited websites from the global history. Links are sorted by time, similar to Firefox's *Go* menu. Selecting any of these creates a link from that page to the current page.

PageLinker 0.2 also created a reverse link, from the current page to the one just selected. We reworked the *Shortcut* contextual bookmarks menu to separate links by direction. One list presents links *to* the current page (either via copy-paste or direct selection). The other list presents links *from* the page. Links on both menus were ordered by recency. Based on user requests, we also added the ability to delete a contextual bookmark by right clicking on the corresponding menu item. After one week of use, users said the *link to* menu was too complex and redundant. Bidirectional links presented in two different menus were also too heavy-weight and users did not notice they that could delete them.

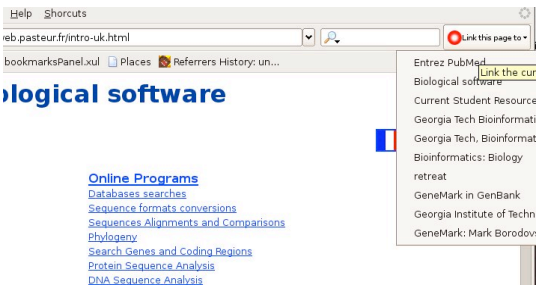


Figure 3: Linking menu prototype (PageLinker 0.2)

PageLinker 0.3 simplified the linking menu to include just the last visited pages. We also classified bidirectional *Shortcuts* by order of recency. Finally, we integrated contextual bookmarks and linking via the bookmarks sidebar (Figure 4). Most users quickly began using the bookmark sidebar instead of the menu. They found it useful to have their contextual bookmarks visible immediately upon changing pages, without needing to click on the menu list, since contextual bookmarks change from one page to the next.

We used PageLinker 0.3 for the field experiment (described in the next section). After the experiment, we released PageLinker 1.0 which included a minor modification: To avoid confusion between the contextual bookmarks list and the linking list, we converted the *link to* list into a menu. Table 1 summarises the four versions of PageLinker, including the types of links, how contextual bookmarks are created and how to access PageLinker.

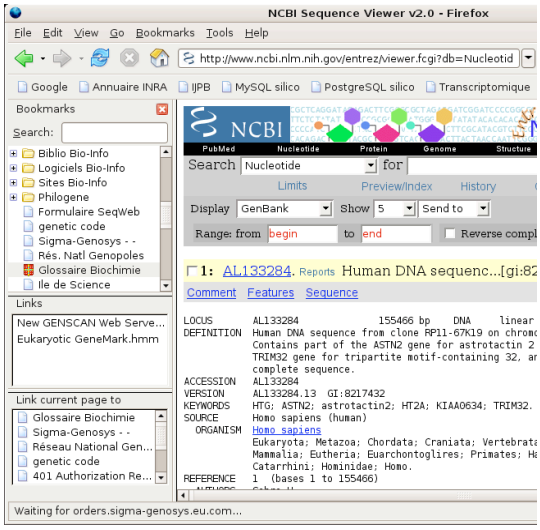


Figure 4: First side-bar prototype.
PageLinker 0.3 presents links to the left of the main window.

Ver	Link type	Creation	Access
0.1	directed, not suppressible	Cut/copy paste	Menubar popup
0.2	bidirectional, suppressible	Cut/copy paste Menubar popup (open & last 7 pages)	Menubar popup
0.3	bidirectional, suppressible, always visible	copy/cut-paste list last-visited pages in Bookmark sidebar	Bookmark list via sidebar
1.0	bidirectional, suppressible, always visible	copy/cut-paste popup via Bookmarks sidebar shows last visited pages	Bookmark list via sidebar

Table 1: Four versions of PageLinker .

EVALUATION

Evaluating history-based tools such as PageLinker poses interesting methodological challenges with respect to validity [10]. We considered the following possibilities:

1. A **laboratory experiment** is easiest to control but poses external validity problems. Our fieldwork indicated that biologists' navigation and bookmarking behavior on unfamiliar tasks with artificial data might differ greatly from their behavior with familiar data and resources, making the results potentially meaningless. Also, users cannot fully leverage their personal knowledge in a lab experiment nor take advantage of their episodic memory. We are also interested in gathering realistic adoption and usage data for PageLinker: not only measuring performance advantages, if any, but also observing how user behavior evolves over time and whether users make the tool part of their repertoire.
2. An **uncontrolled field study** has greater external validity but is very difficult to control. Longitudinal field studies require extensive logging and extensive data analysis, especially if the participants' environment is not modified. Long-term monitoring also raises serious privacy issues and risks interfering with biologists' confidentiality agreements. For example, some biologists asked us to stop recording during the interview if they thought we might see confidential data. These biologists would not have been willing to participate in long-term automatic recording of their activities. Biologists also alternate between periods of intense on-line data analysis and periods of laboratory research. At any point in time, individuals may be out of phase with each other, depending upon who is writing a paper, running an experiment, or analysing data. This diversity complicates any comparisons and analysis of activity logs. For example, it would be difficult to tell, for any one subject, whether a decrease in pages visited was due to PageLinker or an overall change in research activity. It would also be difficult to compare people who were at different phases in their work.
3. A **limited time-series field experiment (or quasi-experiment [10])** offers the optimal compromise, with the external validity of a field study and most of the control offered by a laboratory experiment. Because we wish to compare PageLinker's navigation performance to existing browsers, it makes sense to alternate PageLinker with the user's usual browser. This allows us to track changes in use over time, based on realistic tasks performed in the user's real work setting, together with their existing bookmarks and other revisitation techniques. We chose this third option to evaluate PageLinker.

Method

Participants

Twelve biologists or bioinformaticians (9 men and 3 women between 20 and 40 years old) working in four research institutes (Institut Pasteur, Génopole, Université Paris 5, INRA) participated in the study. All were Firefox users with browsing and bookmarking experience. Two had also participated during the participatory design phase. (Post hoc analysis did not show significant differences between their results and those of other study participants.)

Apparatus

Hardware: Participants used their usual browser with their own bookmarks and history, on their own system: 5 Mac OS X users, 4 Windows users and 3 Linux users.

Software and logging: We used PageLinker 0.3 in the experiment and Navtracer[3] [21], a standard Firefox extension that logs user interactions with the browser, to record user activity in both conditions. Navtracer runs on any version of Firefox (from 1.0 to 2.0) and could be installed and disabled rapidly in each user's browser without requiring special knowledge. This allowed us to minimise disruptions and let participants continue using their standard bookmarks, history and other Firefox extensions in both conditions in the evaluation.

To protect privacy, the extension does not begin logging automatically. Rather, users press a *start* button added to a Firefox window and fill out a form describing the experimental condition. This gives users full control of logging: they can pause, resume or stop at any time. When Navtracer was first installed, we showed users how to enable and disable logging and where the CSV log file was stored. They were invited to delete the file or modify its contents if they had concerns about what had been logged.

The extension registers various event handlers to detect the opening or closing of tabs and windows and the acquiring or loss of focus. It also tracks web-page changes and the relations between them via the page referrer. Switches between documents (windows or tabs) are also recorded. Event handlers append log data to a plain text file stored in the user's profile folder. Timestamps are systematically added to every record. Navtracer also logged PageLinker events such as link creation and usage of created links.

Scenario design

The experiment scenario was based on our observations of common tasks and navigation patterns, including:

- *Search:* web search engines, biological databases, directories
- *Parallel exploration:* same analysis with two programs
- *Results comparison:* same analysis with two programs
- *Analysis:* visual scan of results to check validity and pertinence
- *Biological links directory:* scanning for options
- *Repeated path:* access the same page

We created a scenario with five related subtasks (Figure 5) with the aid of two biologists from the same environment. The scenario had to be short enough (between 15 and 20 minutes) so that it would not be too time consuming for participants, but still be representative of their tasks and understandable for every specialty. The five tasks illustrate aspects of web navigation presented above. The scenario is open and participants were encouraged to use their usual websites to perform the tasks. The websites presented here were the most commonly used, taken from different servers to illustrate the resource diversity faced by biologists. The five tasks are:

1: Database search: Find the gene corresponding to *human muscular dystrophy* and choose the nucleotide sequence attached to the TRIM32 gene (usually used NCBI[4]).

2,3: Parallel exploration: Analyse the nucleotidic sequence with two different tools, e.g., Genscan[5] and Genemark[6], to predict what the peptide sequence would be.

4: Comparison: Compare the two predicted sequences, e.g., using bl2seq[7] to check if predictions are reliable (result R1).

5: Analysis and visual scan: Analyse one of the predicted peptide sequences to find regions of local similarity with other sequences with Blast[8] (result R2). The goal is to find species other than homo sapiens that express the same protein with a high degree of confidence and are interesting for researchers looking for a related analysis or literature.

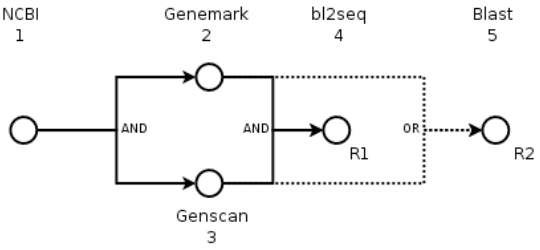


Figure 5: Scenario structure: Task 1 is performed first, followed by tasks 2 and 3 which are often performed in parallel. Task 4 is possible only after tasks 1-3 are complete and produces R1. Task 5 may be conducted independently after tasks 2 or 3 and produces R2.

Procedure

We used an ABAB within-subjects design, with one factor:

FireFox: Firefox browser with logging
PageLinker: Firefox browser with logging and PageLinker

[3] http://navtracer.mozdev.org/

[4] http://www.ncbi.nlm.nih.gov/
[5] http://genes.mit.edu/GENSCAN.html
[6] http://exon.gatech.edu/GeneMark
[7] http://bioweb.pasteur.fr/seqanal/interfaces/bl2seq
[8] http://www.ncbi.nlm.nih.gov/BLAST/Blast.cgi

Users alternated between the PageLinker and the unmodified Firefox conditions at one-week intervals. Users kept their history, standard bookmarks and other Firefox extensions when changing conditions. This allowed them to work with their own real data settings instead of an empty initialized browser or one with artificial bookmarks and history the user was not familiar with.

Our goal was to collect data over long periods without extensive logging, so we sampled their navigation by taking a snapshot of the state of their bookmarks and asking them to follow the five-task scenario described above. Full counterbalancing of tasks across subjects is impossible, because PageLinker requires a first visit to websites to create the contextual links. (In other words, the unmodified Firefox condition must be run first, for all subjects.) We used an ABAB procedure, repeating each condition twice, to dissociate learning effects as much as possible from improvements due to PageLinker.

During the evaluation, each session was separated from the next by an interval of at least a week. Based on our previous observations, it appeared that seven days, including a week-end, should be long enough for participants to partially forget the exact details of what they had done during the previous session. This reduced the learning effect and is also representative of biologist's typical behavior: They frequently perform a series of tasks for one purpose and then repeat it after days or weeks of performing other tasks.

One experimenter visited each of the participants in their lab once a week for a month. During each visit, participants were asked to perform the same scenario. In the first session, we introduced PageLinker and invited the biologists to use it freely until they felt comfortable with link creation and use. This training period lasted between 10 and 15 minutes. We first showed participants how to create links either by copy/paste or the menu list. They were then free to try creating lists between any pages they liked. We finally asked them to determine pages they thought were related to each other and to create links between them using the two techniques. In case they had no idea of what to link, we suggested that they create links between pages they had visited during a recent break so as to avoid conflicts with our scenario. (Note: This occurred primarily during the first session, with a few biologists who had not done this type of analysis for a long time.)

The experimenter then presented the standard scenario, explaining its biological purpose and the necessary steps to achieve it. During this phase, we avoided mentioning any particular online tools and encouraged participants to use their favorite applications, portals or search engines. Our only guidance consisted of reminding them of the next task after they completed the previous one. Tools and portals were only suggested if they did not know what software was appropriate for a task or if their usual application server was down. (Note: The server went down twice in the course of the month-long study and ran very slowly approximately once per participant.)

The PageLinker extension remained installed during all phases of the study, but was invisible to users during the Firefox-only conditions. In the latter case, it simply logged the creation of links between pages via copy/paste, as a conventional history tool. To protect privacy, we disabled the logging extension after each session. We also asked users if they wanted PageLinker to be disabled between sessions: All decided to keep it. To avoid interference between contextual bookmarks created during the experiment and non-experiment phases, we stored the contextual bookmarks in different files.

Predictions and Hypothesis

Based on feedback from our first field release and our personal use of the extension, we predicted the following results: We predicted that PageLinker would generate fewer pageloads and fewer clicks per task and reduce time spent on each task. We also predicted that with PageLinker, the majority of links would be created on the first visit to each relevant website. Since we had interacted with the users and iteratively responded to their requests during the design of the tool, we also expected our participants to be mostly satisfied with the design and interaction techniques used in the main experiment.

RESULTS AND DISCUSSION

Quantitative Results

PageLinker performed significantly better than the unmodified Firefox browser with respect to the following dependent variables (Figure 6):

- task completion time was 28% shorter ($p<0.01$)
- 22% fewer clicks occurred ($p<0.01$)
- 38% fewer pages loaded ($p<0.01$)

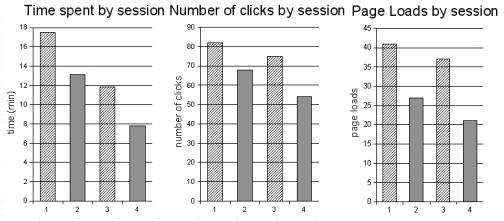


Figure 6: Evolution of time, clicks and page loads over sessions. Columns 1 & 3 are Firefox only, columns 2 & 4 are PageLinker.

If we focus more specifically on the limited time series, we observe the same pattern for clicks and page loads, although the difference is only significant for the number of page loads. The decreased number of page loads corresponds to the biologist seeing 38% ($p<0.05$) fewer pages during a typical day. Although there is an overall learning effect, i.e. biologists become more efficient running the tasks in the scenario over time, there is also a strong effect of

PageLinker. Columns two and four (PageLinker conditions) are always more efficient than columns one and three (Firefox-only conditions).

The overall number of links created is not significantly different over the four sessions: A mean of 20 contextual bookmarks were created during the first session and 12 during each other session. Participants never had too many contextual bookmarks, with the corresponding risk of overload. This is because the use of contextual bookmarks increases linearly with the number of created links $F_{1,11}= 8.73$, ($p<0.05$). In summary, these results suggest that PageLinker actively facilitates page revisitation:

- Fewer page loads shows that users visited fewer search websites and transition pages,
- Fewer clicks shows they used fewer transition pages, and
- Fewer pages seen shows they took les time to complete the five tasks of the scenario.

Qualitative Results

The participants' use patterns in the Firefox-only condition were very similar to those we saw in the earlier design phase. For example, they used directories of biological resources to find links to on-line programs and said that they usually preferred to use search engines to find a link, even when they knew that they had a bookmark for that particular page.

We observed several ways that PageLinker assisted users in their work flow. When interruptions occurred during the evaluation, such as people asking questions, coffee breaks, and phone calls, PageLinker helped them reorient themselves when they returned to their task. By seeing the links to and from the pages, participants could more easily remember what they were doing and what their goals had been. We also observed that it helped users in case of server slowdown or breakdown. They began to keep alternate links to the same program on different servers, something they never did with standard bookmarks because it would have generated an unacceptably large number of bookmarks. Unlike automation tools, PageLinker is robust to changes in internal page structure. The simplicity of our solution allows easy re-linking whenever a website's structure changes.

Limitations of the Experiment

Dissociating PageLinker effects from learning effects is complex when interpreting the time spent on the scenario and the number of clicks. Time is highly correlated with external factors, such as the current server load. For example, users may wait more than five minutes for a Blast result from the NCBI if the servers are heavily loaded.

Another potential problem is assessing the correlation between the number of contextual bookmark links and their use. Perhaps a month-long evaluation is too short to overload the contextual bookmarks menu. We expect that the recency classification we use, which only shows the most recently used links, should reduce the overload effect, but we would need a much longer study to find out.

Finally, PageLinker can only reduce hyperlinks clicks, not the clicks needed to fill in forms. Nevertheless, the logger counted all clicks indiscriminately, whether they occurred on links or on forms. PageLinker thus accounted for only a small percentage of the overall number of clicks and the reduction was indistinguishable from noise.

Three-Month Follow-up

After the evaluation, we released PageLinker 1.0 which modified how contextual bookmarks are created. Figure 7 shows that the *link to* list has been changed into a menu.

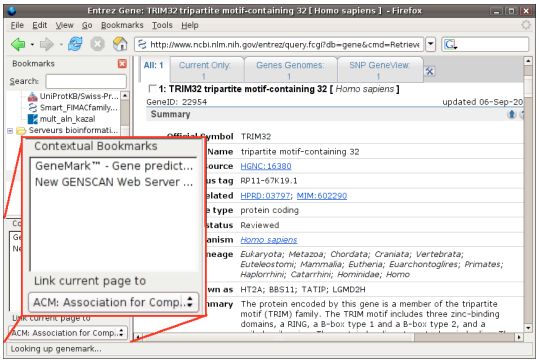


Figure 7: PageLinker version 1.0.

Three months later, we sent the participants a questionnaire (Table 2). Of the 12 participants, two had changed institution and did not answer, two had changed browser or workstation without re-installing PageLinker and eight still used PageLinker. The key questions in the questionnaire are presented in Table 2:

Question	Mean	SD
How usable is the link creation?	4.33	0.87
How usable are the created links?	4.44	0.73
How useful are the links created?	3.56	1.24

Table 2: Responses to the questionnaire using a five point Lickert scale: 1 = *not at all*, 5 = *very*.

Participants reported two primary uses of PageLinker in the months following the field experiment. The first is similar to that described in our scenario, in which users create chains of web pages, applying results from one page to subsequent forms. The second, more frequent PageLinker use involves creating relations between web pages that the users navigate frequently. We call this fuzzy grouping: the pages are related to each other without the hierarchical order imposed with regular bookmarks or other link organisers. If the user's area of interest changes slightly and they visit new sites, they simply add a few links to the

pages already linked to them and forget about obsolete links.

CONCLUSION

We began by addressing a specific problem faced by biologists: linking data output pages to data analysis pages. After we released the first version of PageLinker, users appropriated it, thus revealing the need for a more general contextual bookmark tool. Users sought ways to associate pairs of web pages and thus facilitate future navigation within groups of previously visited pages. Our studies with biologists demonstrated that PageLinker's contextual bookmarks improve web page revisitation and that, unlike history and bookmarks mechanisms, they are less prone to information overload. The philosophy of letting users handle their links allowed the tool to be both simple and robust to changes in Web content and user practices. Three months after the study, at least two-thirds of the participants were still using PageLinker.

Contextual bookmarks display links to other pages relevant to the user, depending upon the web page visited. Study participants considered these links both easy to create and to use. They also found them easy to understand and predict, since relevance is not decided by an automatic process but by users' explicit actions.

Future work includes the development of visual cues to indicate where users are in their navigation, presenting previously seen pages before and after the one being displayed on screen. Users also expressed the desire to share their links with others in their research team.

Biologists are heavy users of web browsers and are thus a good target audience to study when exploring navigation problems. Yet, the concepts developed for PageLinker are more general and likely to apply to a wide variety of users. In his classic article, “As We May Think”, Vannevar Bush [4] argues that the human mind operates by association, connecting items into a web of trails. In the spirit of his Memex idea, we offer a tool that allows users to “build a trail of interest through the maze of materials available”. Linking web resources while navigating is a powerful way to find information again and to reflect the users' thinking as they explore.

ACKNOWLEDGMENTS

We would like to thank all the biologists who volunteered their time for interviews, workshops and the study. Special thanks to Sophie Créno and Florence Hantraye in particular, for their help creating the scenarios, as well as Samira Laribi and Bertrand Neron for their generous explanations of biological research and processes. Partial funding was obtained from the MicroMégas project.

PageLinker is available at: http://rabidfox.mozdev.org

REFERENCES

1. D. Abrams, R. Baecker, and M. Chignell. Information archiving with bookmarks: personal web space construction and organization. In *CHI’98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41-48, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.
2. V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating Web navigation with the WebVCR. *Computer Networks*, 1:503–517, 2000.
3. S. Baker. Mosaic-surfing at home and abroad. In *SIGUCCS ’94: Proceedings of the 22nd annual ACM SIGUCCS conference on User services*, pages 159–163, New York, NY, USA, 1994. ACM Press.
4. V. Bush. As we may think. *Atlantic Monthly*, 176(1):101–108, June 1945.
5. T. B. C. Sorensen, D. Macklin. Navigating the world wide web: Bookmark maintenance architectures. *Interacting with Computers*, 13:375–400, 2001.
6. L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the world-wide web. In *Proceedings of the Third International World-Wide Web conference on Technology, tools and applications*, pages 1065–1073, 1995. Elsevier North-Holland, Inc.
7. A. Cockburn, S. Greenberg, S. Jones, B. McKenzie, and M. Moyle. Improving web page revisitation: Analysis design and evaluation. *IT Society*, 1:159–183, 2003.
8. A. Cockbum, S. Greenberg, B. McKenzie, M. Smith and S. Kaasten. *WebView: A graphical aid for revisiting Web pages*. Proc OZCHI'99, Australia, 1999.
9. A. Cockburn and B. J. McKenzie. What do web users do? an empirical analysis of web use. *Int. J. Hum.-Comput. Stud.*, 54(6):903–922, 2001.
10. T. Cook and D. Campbell. *QuasiExperimentation: Design & Analysis Issues for Field Settings*. Houghton Mifflin Company, 1979.
11. J. Fujima, A. Lunzer, K. Hornbaek, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. *In UIST ’04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, 2004.
12. R. Hightower, L. Ring, J. Helfman, B. Bederson, and J. Hollan. Graphical multiscale web histories: A study of PadPrints. In Hypertext '98 Proceedings, pages 58--65, 1998.
13. W. Jones, H. Bruce, and S. Dumais. Keeping found things found on the web. *In CIKM ’01: Proceedings of the tenth international conference on Information and knowledge managemen*t, pages 119–126, New York, NY, USA, 2001. ACM Press.

14. C. Letondal. A web interface generator for molecular biology programs in unix. *Bioinformatics*, 17, 2001.
15. C. Letondal and W. E. Mackay. Participatory programming and the scope of mutual responsibility: balancing scientific, design and software commitment. *In PDC 04: Proceedings of the eighth conference on Participatory design*, pages 31–41, New York, NY, USA, 2004. ACM Press.
16. W. Mackay, C. Letondal, G. Pothier, K. Bøegh, and H. Sørensen. The missing link: augmenting biology laboratory notebooks. In *UIST 2002*.
17. W. Mackay. *Users and Customizable Software: A Co-Adaptive Phenomenon*. PhD thesis, M.I.T., 1990.
18. W. Mackay. Triggers and barriers to customizing software. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 153–160, New York, NY, USA, 1991. ACM Press.
19. W. Mackay and A. L. Fayard. Video brainstorming and prototyping: techniques for participatory design. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 118–119, New York, NY, USA, 1999. ACM Press.
20. J. Nielsen. User interface directions for the web. Commun. ACM, 42(1):65–72, 1999.
21. N. Roussel, A. Tabard, and C. Letondal. All you need is log. *In WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, May 2006. 4 pages.
22. Stevens, Goble, Baker, and Brass. A classification of tasks in bioinformatics. *BIOINF: Bioinformatics*, 17, 2001.
23. Suchman, L. (1987) *Plans and Situated Actions.* Cambridge, England: Cambridge University Press.
24. L. Tauscher and S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies*, 47(1):97–137, 1997.
25. L. Tauscher and S. Greenberg. Revisitation patterns in World Wide Web navigation. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'97*, 1997.
26. J. Teevan, C. Alvarado, M. Ackerman, and D. Karger. The Perfect Search Engine Is Not Enough: A Study of Orienteering Behavior in Directed Search. In *Proceedings of ACM CHI 2004 Conference on Human factors and Computing Systems*, pages 415–422. ACM Press, Apr. 2004.
27. H. Weinreich, H. Obendorf, E. Herder, and M. Mayer. Off the beaten tracks: exploring three aspects of web navigation. *In WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 133–142, New York, NY, USA, 2006. ACM Press.