Support informatique à une communication médiatisée

Nicolas Roussel

UNIVERSITE PARIS-SUD

U.F.R. SCIENTIFIQUE D'ORSAY

THESE
présentée
pour obtenir

le grade de DOCTEUR EN SCIENCES
de L'UNIVERSITE PARIS-SUD XI

par

Nicolas Roussel

Sujet :

Support informatique à une communication médiatisée

Soutenue le 17 juillet 2000 devant la Commission d'examen :

M. Michel Beaudouin-Lafon (Directeur de thèse)

Mme Joëlle Coutaz (Rapporteur)

M. Alain Derycke (Rapporteur)

Mme Marie-Claude Heydemann

M. Claude Kintzig

Remerciements

Michel Beaudouin-Lafon est un directeur de thèse passionné qui ne recule devant rien : il n'a pas hésité à mettre en pratique notre intérêt commun pour le travail collaboratif à distance en allant s'installer au Danemark au milieu de cette thèse. Cette passion est par bonheur contagieuse et ces années passées à travailler à son contact, même à distance, ont été très enrichissantes. Merci Michel pour tes idées, tes conseils, et ton aide.

Merci à France Télécom R&D pour le soutien financier dont ce travail a bénéficié. Merci en particulier à Georges Buchner et Claude Kintzig pour l'intérêt qu'ils y ont porté. Merci également au consortium ERCIM qui m'a permis de terminer cette thèse au GMD et grâce auquel je vais pouvoir poursuivre mes travaux à l'Université de Fribourg.

Merci à Alain Derycke et à Joëlle Coutaz pour le temps si précieux qu'ils m'ont accordé en acceptant d'être rapporteurs de cette thèse. Merci également à Marie-Claude Heydemann d'avoir bien voulu participer à mon jury.

Merci à Wendy, Mountaz, Thomas, Jean-Daniel, Christophe, Philippe et Catherine pour les discussions passées et celles que nous avons encore. Merci surtout à Stéphane et Paul avec qui j'ai pu prolonger tard le soir ces discussions et bien d'autres.

Merci à Agnès, Philippe, Anne et Daniel qui m'ont donné le goût des études, de l'Informatique, des voyages, des langues étrangères et du Jazz. Tout cela m'a bien aidé. Cette thèse vous est dédiée.

Enfin, muito obrigado à Rosane qui a su m'encourager et me motiver quand il le fallait, qui a accepté de "vivre" dans un mediaspace pendant trois ans, et qui met chaque jour un peu plus de Samba et de Bossa Nova dans ma vie. Cette thèse est également pour toi.

Table des matières

REMERCIEMENTS 2

TABLE DES MATIÈRES 3

TABLE DES ILLUSTRATIONS 6

INTRODUCTION 9

Avant-propos : l'Interaction Homme-Machine 9

Sujet de cette thèse 11

Structure du mémoire 12

CHAPITRE I USAGES DE LA VIDÉO DANS LES SYSTÈMES INTERACTIFS 14

1. Situations formelles : visiophone, visioconférence et téléréunion 15

2. L'informel et le quotidien 20

3. Partage de vues et fusion des espaces 23

4. Traitement d'images pour l'interaction homme-homme ou homme-machine 27

5. Résumé du chapitre 32

CHAPITRE II MEDIASPACES 33

1. Du Media Space aux mediaspaces 33

2. Spécificité des mediaspaces 42

3. Caractéristiques fondamentales des mediaspaces 47

4. Résumé du chapitre **55**

CHAPITRE III LE WEB COMME INFRASTRUCTURE LOGICIELLE POUR LES MEDIASPACES **57**

1. Choix de l'infrastructure : une affaire de public **57**

2. Le Web **59**

3. Le Web comme infrastructure logicielle pour les mediaspaces **63**

4. Résumé du chapitre **70**

CHAPITRE IV MEDIASCAPE ET VIDEOSERVER **72**

1. Mediascape **73**

2. VideoServer **81**

3. Usages de videoServer **86**

4. Notification, contrôle et adaptation du contenu **92**

5. Résumé du chapitre **98**

CHAPITRE V AU-DELÀ DU NAVIGATEUR : NOUVEAUX CLIENTS ET NOUVEAUX PROTOCOLES **100**

1. VideoClient **100**

2. Nouveaux protocoles pour videoServer **107**

3. Mise en œuvre d'un système de communication basé sur videoServer **114**

4. Résumé du chapitre **118**

CHAPITRE VI VIDEOSPACE **120**

1. Motivations **120**

2. La boîte à outils videoSpace **122**

3. Interface de programmation de videoSpace **123**

4. Applications de base proposées par videoSpace **133**

5. Discussion **134**

6. Résumé du chapitre **136**

CHAPITRE VII PROTOTYPAGE DE NOUVEAUX USAGES DE LA VIDÉO AVEC VIDEOSPACE **137**

1. La main comme télépointeur **137**

2. Vers la téléconvivialité **149**

3. L'environnement informatique vu comme une source d'images **157**

4. Résumé du chapitre **165**

CONCLUSION **167**

Résumé de la contribution **167**

Limites et perspectives **169**

ANNEXE A – ACQUISITION VIDÉO EN TEMPS RÉEL **171**

1. Acquisition vidéo sous SGI IRIX **172**

2. Acquisition vidéo sous Linux **173**

ANNEXE B – MINISERVER.CXX **175**

ANNEXE C – VIDEOCLIENT.CXX **177**

BIBLIOGRAPHIE **179**

Table des illustrations

Figure 1. Le Picturephone dans 2001 l'odyssée de l'espace 16
Figure 2. Salle aménagée pour la visioconférence 17
Figure 3. Système mobile (Compression Labs Incorporated, 1997) 18
Figure 4. CU-SeeMe (White Pine Software, 1997) 19
Figure 5. ShareVision (Creative Technology, 1997) 19
Figure 6. Hole-in-Space (Electronic Cafe) 21
Figure 7. Espaces publics du Media Space (extrait de [BHI93]) 22
Figure 8. L'un des bureaux du Media Space (extrait de [BHI93]) 22
Figure 9. Configuration matérielle de VideoDraw (extrait de [TM91]) 23
Figure 10. Configuration matérielle de TeamWorkStation (extrait de [Ish99]) 25
Figure 11. Exemple d'utilisation de TeamWorkStation (H. Ishii) 25
Figure 12. La métaphore de ClearBoard (H. Ishii) 26
Figure 13. Configuration matérielle de ClearBoard (extrait de [IKG93]) 26
Figure 14. ClearBoard (extrait de [IKG93]) 27
Figure 15. Videoplace (Ars Electronica Center) 29
Figure 16. Utilisation du geste dans Videoplace (Ars Electronica Center) 29
Figure 17. Digital Desk (extrait de [Wel93]) 30
Figure 18. Principe de la fenêtre virtuelle 31
Figure 19. Configuration typique d'un mediaspace analogique. Les équipements audio sont omis dans un souci de simplicité 35
Figure 20. Configuration typique d'un nœud de mediaspace analogique. 35
Figure 21. Vue d'ensemble de CAVECAT (Ontario Telepresence Project) 37
Figure 22. Interface de RAVE (extrait de [BM90]) 38
Figure 23. Classement des services par degré croissant d'engagement attendu 39
Figure 24. Fondu d'images pour annoncer un coup d'œil (extrait de [TR94]) 40
Figure 25. Panneau de contrôle de RAVE pour le contrôle et la notification liés au service de coup d'œil. L'utilisateur est en train de choisir la liste des personnes autorisées à demander ce service. 41
Figure 26. Interface de CAVECAT (Ontario Telepresence Project) 41
Figure 27. CorelVideo (Corel Corporation, 1997) 42
Figure 28. Une partie du câblage de CAVECAT (Ontario Telepresence Project) et l'AV Suite de RAVE (Xerox EuroPARC). Le panneau à côté des câbles indique : "Do not touch any of these wires" 43
Figure 29. Intégration des moyens audiovisuels du mediaspace (Ontario Telepresence Project) 49
Figure 30. Intégration des moyens de contrôle du mediaspace (Ontario Telepresence Project) 50
Figure 31. Schéma d'URL HTTP absolue 61
Figure 32. Format d'une requête HTTP 62
Figure 33. Format d'une réponse HTTP 62
Figure 34. Exemple de code HTML 62
Figure 35. Exemple de présentation du document décrit par le code HTML de la Figure 34 63
Figure 36. Principe de la transmission d'une séquence d'images JPEG utilisant le mécanisme de server-push 66
Figure 37. " On the Internet, nobody knows you're a dog " (dessin de P. Steiner paru dans The New Yorker le 5 juillet 1993) 68

Figure 38. Exemple d'informations concernant un utilisateur distant obtenues à partir d'une connexion TCP 69
Figure 39. Exemple d'informations additionnelles transmises par un client dans une requête HTTP 69
Figure 40. Configuration matérielle de Mediascape 73
Figure 41. Formulaire d'identification de Mediascape 74
Figure 42. Traitement d'une requête `call` par le serveur de Mediascape. 75
Figure 43. Traitement d'une requête `grab` par le serveur de Mediascape 76
Figure 44. Composition d'un PostIt Mediascape dans un navigateur Web 76
Figure 45. Exemple de PostIt affiché. En demandant le réaffichage du message par le navigateur, le destinataire peut mettre à jour l'image de l'auteur du message. En cliquant sur celle-ci, il peut établir une liaison audio/vidéo avec lui. 77
Figure 46. Interface de base de Mediascape. Conversy a fermé sa porte et celle de Roussel est entrouverte. Les deux icônes de droite donnent accès à des sources vidéo publiques, un magnétoscope et une station de travail 78
Figure 47. Coordination entre deux coauteurs par un document augmenté par Mediascape. Ici, nous pouvons voir que les deux auteurs sont présents en même temps 80
Figure 48. Intégration des services de Mediascape dans un outil de messagerie électronique. L'image affichée est capturée au moment de la lecture du message. 80
Figure 49. Document HTML montrant plusieurs videoServers. Les images sont fixes par défaut, mais se transforment en flux vidéo au passage de la souris. 83
Figure 50. Utilisation de JavaScript pour ajouter une "fenêtre vidéo" flottante sur un document HTML 83
Figure 51. Dégradation de la qualité de l'image par augmentation du taux de compression JPEG 85
Figure 52. Exemples d'images typiques de videoServer 87
Figure 53. La même image, présentée à des résolutions de 128x96, 80x60 et 64x48 89
Figure 54. Succession d'images 80x60 89
Figure 55. Contrôle d'une caméra par le Web : les icônes permettent de changer l'orientation et le zoom de la caméra, les liens sous l'image permettent de changer sa résolution 90
Figure 56. Avant et après la notification : communication mimo-gestuelle répondant à la notification du videoServer 91
Figure 57. Utilisation de videoServer avec une applet Java de dialogue textuel 92
Figure 58. Exemple de description de document pour myWebServer. Le document utilise la feuille de style `html` et définit une nouvelle macro (`list`). 95
Figure 59. Exemple de description pour myWebServer d'un énoncé avec notes pour l'enseignant et correction 96
Figure 60. Exemple de description d'un article scientifique incluant des mécanismes de contrôle d'accès et de notification 97
Figure 61. VideoClient 101
Figure 62. Description du type MIME `video/x-videoSpace` pour son utilisation par le navigateur Netscape 102
Figure 63. Exemple de script de liaison entre le navigateur et videoClient 102
Figure 64. Principe du fenêtrage X Window. La fenêtre «racine» correspond au fond de l'écran. Ce que l'utilisateur perçoit comme étant l'application est en fait composé de trois fenêtres : les décorations, la fenêtre principale et le bouton. 103
Figure 65. l'application talk augmentée par videoClient 104
Figure 66. Script Shell UNIX réalisant le talk augmenté 104
Figure 67. Code Tcl permettant d'insérer un videoClient dans un container Tk 105
Figure 68. Intégration de videoClient dans une application Tcl/Tk 105
Figure 69. Intégration de plusieurs videoClient dans une application Tcl/Tk de type vue d'ensemble 106
Figure 70. L'application groupdraw "augmentée" par l'ajout de liaisons vidéo entre les participants 106
Figure 71. Utilisation de HTTP et UDP dans le cadre d'une liaison par VSTP : la requête est transmise par HTTP, les images par UDP 111
Figure 72. Image typique de videoServer codée en JPEG à une résolution de 160x120 112
Figure 73. Compression JPEG sur des images caractéristiques de videoServer et bande passante nécessaire pour la transmission de 15 images par seconde 112
Figure 74. Principe de la diffusion d'images de videoServer par UDP multicast 114
Figure 75. Exemple d'affichage de multiVideoClient. Les images montrent les prévisions météorologiques pour la région, la cafétéria et l'espace de détente. 115
Figure 76. Exemple de script d'initialisation de multiVideoClient 116
Figure 77. Liaisons utilisées entre les videoServers et les multiVideoClients pour relier la cafétéria et l'espace de détente 116

Figure 78. Transmission unidirectionnelle d'un flux vidéo PAL par VSMP entre deux O2, de 22h à minuit 117
Figure 79. Transmission bidirectionnelle de 2 images 320x240 par seconde entre une O2 et une Octane 117
Figure 80. Exemple d'utilisation des outils du projet MASH pour du télé-enseignement (Berkeley MASH Project) 121
Figure 81. Modèle simpliste de l'utilisation de la vidéo dans des applications interactives 122
Figure 82. Cycle de vie d'une image dans videoSpace 124
Figure 83. Exemple de filtre de transformation 126
Figure 84. Exemple de filtre d'analyse 126
Figure 85. Exemples de transformations successives d'une image. La première transformation change l'adresse des données, la seconde ne change que le contenu 128
Figure 86. Affichage de flux videoSpace dans Inventor 129
Figure 87. Les curseurs standards du système X Window 138
Figure 88. Avant et après le procédé de chroma-keying 139
Figure 89. Superposition d'un flux vidéo détouré sur un fond fixe 140
Figure 90. Après un changement de taille, de position et du niveau transparence 141
Figure 91. Contrôle du pointeur vidéo par une tablette placée sous le fond uni 143
Figure 92. Second prototype utilisé avec des overlay planes de profondeur 2 et 8 145
Figure 93. Environnement Macintosh accessible par RFB depuis un client VNC sous X Window 147
Figure 94. Principe de fonctionnement de Xvnc 148
Figure 95. Troisième prototype utilisant VNC 148
Figure 96. Vous voulez boire quelque chose ? (extrait de [TB94]) 149
Figure 97. Mass Hallucinations (extrait de [DGH+98]) 150
Figure 98. Liquid Views (GMD, IMK.MARS) 151
Figure 99. L'InteracTable (GMD, IPSI) 151
Figure 100. Le puits : idée générale 153
Figure 101. Le puits : vue subjective 153
Figure 102. Image typique des conditions de prise de vue d'une des caméras d'un puits 154
Figure 103. Fusion et partition, deux exemples de composition de plusieurs sources d'images 154
Figure 104. Les deux premiers prototypes 155
Figure 105. Montage vidéo pour la connexion des deux puits 156
Figure 106. Montage vidéo de la future évolution du puits 157
Figure 107. Exemple de mosaïque vidéo 157
Figure 108. EchoCom (GMD, FIT.MMK) 159
Figure 109. Animation d'un cycle cardiaque transmise par videoSpace vers un client Java et un navigateur Web 160
Figure 110. VideoWorkspace 161
Figure 111. Utilisation de la transparence et des transformations géométriques dans MacOS X (Apple) 163
Figure 112. Exemple d'affichage envisagé pour un nouvel environnement graphique (simulation réalisée avec gimp) 164
Figure 113. Juxtaposition de fenêtres non décorées 164

Introduction

Cette thèse s'inscrit dans la discipline de l'Informatique désignée sous le nom d'*Interaction Homme-Machine*. Avant d'en venir à la présentation du sujet de nos travaux et à la structure de ce mémoire, nous pensons qu'il est bon de donner quelques repères concernant cette discipline qui nous ont guidés tout au long de notre travail.

Avant-propos : l'Interaction Homme-Machine

Le point de départ de cet avant-propos est la définition proposée par le groupe d'étude de l'Interaction Homme-Machine de l'ACM (Association for Computing Machinery). Cette définition regroupe trois notions qui constituent les fondements de cette discipline : les *systèmes interactifs*, l'*utilisateur* et le *contexte d'interaction*.

> «L'Interaction Homme-Machine est une discipline consacrée à la conception, la mise en œuvre et à l'évaluation de systèmes informatiques interactifs destinés à des utilisateurs humains ainsi qu'à l'étude des principaux phénomènes qui les entourent.» [HCI]

Un *système interactif* est un système dont le fonctionnement dépend d'informations fournies par un environnement externe qu'il ne contrôle pas [Weg97]. Les systèmes interactifs sont également appelés *ouverts*, par opposition aux systèmes *fermés* - ou *autonomes* – dont le fonctionnement peut être entièrement décrit par des algorithmes.

L'Interaction Homme-Machine s'intéresse aux systèmes informatiques interactifs contrôlés par des *utilisateurs humains*. Du point de vue de la machine, l'humain a beaucoup de défauts : il est indécis, désordonné, inattentif, émotionnel, illogique [Nor94]. Mais il présente une grande qualité : sa capacité d'adaptation. Cette capacité d'adaptation a longtemps contribuée à une vision du

progrès centrée sur le développement des capacités technologiques de la machine et se désintéressant totalement de sa relation avec l'humain. A quoi bon se préoccuper de lui, puisqu'il s'adapte si bien à tout ce qu'on lui propose ? Le slogan de l'Exposition Universelle de 1933 illustre parfaitement cette vision : «la Science trouve, l'Industrie applique et l'Homme s'adapte».

Le degré d'interactivité d'un système peut se mesurer au nombre et à la nature de ses échanges avec les utilisateurs. On peut ainsi dire que les premiers systèmes informatiques basés sur l'utilisation de cartes perforées et l'allumage de diodes étaient peu interactifs. L'augmentation de la puissance des ordinateurs a depuis permis l'avènement des interfaces graphiques et l'exécution parallèle de plusieurs tâches, deux éléments qui contribuent de façon importante à l'interactivité des systèmes actuels.

Confrontés à cette interactivité croissante des systèmes informatiques que nous utilisons, nous observons aujourd'hui les limites de la vision du progrès centrée sur la machine. La progression constante de la technologie se heurte chaque jour un peu plus au seuil de complexité au-delà duquel notre capacité d'adaptation ne suffit plus. De nombreuses fonctionnalités offertes par les systèmes actuels restent ainsi hors de notre portée[1]. La machine nous paraît alors rigide, inutilement complexe, inadaptée à nos besoins et nous laisse un sentiment de frustration [Bux97].

Nous vivons dans un environnement psychologique, social et culturel complexe. La connaissance de cet environnement par les concepteurs d'un système est donc fondamentale pour pouvoir fournir aux utilisateurs la représentation de ce système et les moyens d'action appropriés. La démarche de conception, de mise en œuvre et d'évaluation des systèmes interactifs doit donc être *centrée sur l'utilisateur* [ND86] et au-delà, sur le contexte d'interaction.

La recherche actuelle en Interaction Homme-Machine s'inscrit dans cette démarche. Elle propose une approche ouverte sur d'autres disciplines comme la Psychologie, la Sociologie, l'Anthropologie ou le Design Industriel pour passer des environnements technocentriques que nous connaissons à une *Informatique située* [Bea00], centrée sur l'utilisateur, le contexte et les usages.

[1] Consultez le manuel de votre magnétoscope ou de votre téléphone pour vous en persuader. Le simple fait de devoir lire ce manuel pour connaître les fonctionnalités de l'appareil est déjà un mauvais signe [Nor96].

Sujet de cette thèse

Héritage des premiers systèmes informatiques, la notion de groupe d'utilisateurs est souvent associée à l'idée de pénurie de ressources et d'accès concurrents qu'il faut organiser ou mieux encore, éviter. De nombreux systèmes actuels fonctionnent encore aujourd'hui selon ce principe et font tout pour cacher et protéger les utilisateurs les uns des autres, leur laissant croire qu'ils sont seuls et que l'ensemble des ressources leur est réservé. L'ordinateur reste donc conçu dans une optique d'usage individuel.

La plupart de nos activités sont pourtant de nature collective, voire collaborative. L'ordinateur nous sert le plus souvent à produire ou organiser de l'information destinée à d'autres personnes, cette information étant ensuite partagée ou échangée avec ces personnes. L'une des conséquences de l'évolution vers une Informatique située est l'intérêt porté à cette nature collective de nos activités et à ses conséquences sur la conception, la mise en œuvre et l'évaluation des systèmes informatiques. Cet intérêt se traduit, depuis une vingtaine d'années, par l'étude des *collecticiels*.

Contrairement aux logiciels traditionnels, les collecticiels ne cherchent pas à isoler les utilisateurs les uns des autres mais à leur faire prendre conscience qu'ils ne sont pas seuls et à leur offrir les moyens nécessaires pour se *coordonner*, *communiquer* et *collaborer*. Les collecticiels regroupent une grande variété d'applications telles que les messageries électroniques, les conférences et réunions assistées par ordinateur, les systèmes d'aide à la décision ou les éditeurs partagés [Kar94].

Le collecticiel est une forme de *communication médiatisée*. Nous utilisons cette expression pour qualifier toute forme d'interaction entre individus effectuée par l'intermédiaire de moyens artificiels. Cette médiation des échanges et du partage de l'information permet de s'affranchir des contraintes de temps et d'espace. Mais dès lors que les individus ne sont plus physiquement dans le même environnement, de nouveaux problèmes apparaissent, liés à la technologie médiatrice. La combinaison de plusieurs technologies, comme l'audio, la vidéo et l'informatique permet bien souvent de contourner ces problèmes, mais l'intégration de ces technologies constitue aujourd'hui encore un véritable défi.

Une grande part de nos interactions sociales est implicite, informelle et repose sur la perception périphérique de la présence et de l'activité des personnes qui nous entourent. Un simple coup d'œil par une porte entrouverte, deux regards qui se croisent ou une rencontre dans un couloir sont des éléments importants pour l'émergence d'une conscience mutuelle qui facilite la coordination et la collaboration. Ces interactions spontanées et opportunistes constituent un deuxième défi pour la conception et la mise en œuvre des environnements de communication médiatisée.

L'objet de cette thèse est l'étude de l'intégration de la vidéo dans les systèmes informatiques interactifs. Nous nous intéressons plus particulièrement à cette intégration dans le contexte de la communication médiatisée, où l'informatique permet à la fois de contrôler des moyens vidéo analogiques et de numériser, transmettre et afficher des flux vidéo numériques. Toutefois, l'informatique permettant de traiter ces flux vidéo numériques, nous nous intéressons également aux utilisations potentielles de ces traitements pour l'interaction entre l'homme et la machine.

Notre travail est centré sur les *usages* de la vidéo que nous voulons reproduire et développer, et non sur la technologie informatique nécessaire au codage et à la transmission des images. Notre but est de fournir les éléments nécessaires pour la conception et la mise en œuvre de systèmes interactifs utilisant la vidéo pour permettre une large variété d'interactions allant de la perception périphérique à des formes de collaboration formelles.

> «Le découvreur est celui qui observe ce qui existe, le comprend et l'explique (…) Les inventeurs, quant à eux, conçoivent et construisent ce qui n'existe pas encore. Ainsi peut-on distinguer, si l'on en ressent l'envie ou le besoin, les deux démarches, scientifique et technologique. En fait, la connexité des deux domaines est si étroite que chacun se nourrit de l'autre et ne progresse qu'en s'appuyant sur l'autre.» [Cur99]

Cette thèse s'inscrit dans la complémentarité des démarches scientifique et technologique : nous voulons comprendre les mécanismes à mettre en œuvre pour permettre une grande diversité des usages informatiques de la vidéo, et nous voulons en même temps fournir les outils qui permettront de développer plus encore ces usages.

Structure du mémoire

Dans le **Chapitre I**, nous présentons différents systèmes commerciaux et travaux de recherche qui illustrent les différents usages actuels de la vidéo dans les systèmes informatiques interactifs.

Parmi les travaux de recherche présentés dans ce premier chapitre figure le Media Space, qui est à l'origine d'une famille d'environnements de communication médiatisée que nous avons choisi d'étudier plus particulièrement. Le **Chapitre II** nous permet d'exposer en détail la spécificité de ces environnements ainsi que les propriétés fondamentales qui les caractérisent.

Nous décrivons et argumentons dans le **Chapitre III** le choix de l'infrastructure logicielle pour la mise en œuvre de nouveaux environnements de type Media Space. Nous montrons comment les langages et protocoles du World Wide Web permettent de construire un système basé sur l'utilisation de serveurs spécifiques et de navigateurs Web standards qui présente les propriétés identifiées dans le chapitre II.

Dans le **Chapitre IV**, nous présentons Mediascape et videoServer, deux prototypes qui mettent en application l'infrastructure décrite dans le chapitre III. Nous décrivons ces deux prototypes, les services qu'ils offrent et les usages qu'ils ont permis de développer. Nous terminons ce chapitre par la présentation d'un serveur de documents expérimental dont la conception est dérivée des services et des usages observés de videoServer.

Le **Chapitre V** est consacré à l'extension des possibilités offertes par videoServer à travers l'utilisation d'applications et de protocoles spécifiques à la place des navigateurs et protocoles Web standards. Nous décrivons ces applications et protocoles et présentons un exemple de leur utilisation combinée.

Les extensions de videoServer présentées dans le chapitre V sont à l'origine de videoSpace, une boîte à outils logicielle pour l'intégration de la vidéo que nous présentons dans le **Chapitre VI**.

Nous décrivons enfin dans le **Chapitre VII** trois ensembles de prototypes réalisés avec videoSpace qui illustrent les possibilités actuelles offertes par cette boîte à outils et qui préfigurent les nouveaux usages de la vidéo qu'elle devrait permettre à l'avenir.

Chapitre I

Usages de la vidéo dans les systèmes interactifs

Le téléphone a été mis au point par Graham Bell en 1876. L'ajout de l'image au son a été envisagé dès les premiers essais de liaisons téléphoniques. Dans *Les cinq cents millions de la Begum*, publié en 1879, Jules Verne décrit ainsi un système de téléconférence de groupe utilisé pour la réunion du conseil de Franceville [BG96]. Dans deux autres de ses œuvres publiées en 1882, *Le château des Carpates* et *La journée d'un journaliste américain*, il décrit également un téléphone alliant l'image au son. A la même époque, dans un ouvrage intitulé *Le vingtième siècle*, Albert Robida décrit le téléphonoscope : un appareil doté d'un écran qui permet à ses utilisateurs de voir un interlocuteur distant mais peut également être utilisé pour assister à des spectacles.

Les prédictions de la fin du dix-neuvième siècle ne se sont pas toutes réalisées. La principale application actuelle de la transmission d'images est la télévision, qui ne correspond qu'à la fonction de diffusion de spectacles du téléphonoscope de Robida. Dans le cadre de cette thèse, nous nous intéressons à son autre fonction, celle qui permet à des personnes de communiquer entre elles grâce à l'image. Nous décrivons dans ce premier chapitre un certain nombre de systèmes existants qui, comme le téléphonoscope, utilisent des flux vidéo pour la communication médiatisée. Ces flux vidéo étant de plus en plus souvent transmis de façon numérique, nous nous intéressons également aux traitements d'images rendus possibles par cette numérisation et à leurs applications pour la communication médiatisée et l'interaction homme-machine.

1. Situations formelles : visiophone, visioconférence et téléréunion

Le 7 avril 1927, la compagnie américaine des téléphones et télégraphes (AT&T) effectua l'une des premières transmissions simultanées d'images et de son entre New York et Washington en utilisant du matériel de télévision et une liaison radio [TB94]. Trois ans plus tard, le 9 avril 1930, le même type d'équipement fût utilisé pour une liaison cette fois bidirectionnelle entre les laboratoires Bell de AT&T et le siège de la compagnie, tous deux situés à New York. Les laboratoires de AT&T poursuivirent l'expérimentation de la transmission d'images jusqu'en 1956, où ils disposèrent enfin d'un système pouvant utiliser les lignes téléphoniques traditionnelles. Ce premier système était très rudimentaire et ne permettait que la transmission d'une image toutes les deux secondes. Après quelques efforts supplémentaires, AT&T aboutit en 1963 au premier visiophone expérimental : le *Picturephone*.

1.1. Le Picturephone

Le Picturephone était capable de transmettre un signal vidéo noir et blanc sur l'équivalent d'une centaine de lignes téléphoniques classiques. Le son était transmis séparément. La caméra et les haut-parleurs étaient montés au-dessus et sur les côtés d'un écran à tube cathodique et un microphone mobile assurait la prise de son. L'ensemble était conçu pour que sa taille soit à la fois compatible avec une utilisation dans un bureau et à la faible résolution que la technique utilisée pouvait transmettre (250 lignes). L'écran était donc petit, 7 pouces de diagonale, et les images un peu floues. De nombreux contrôles permettaient toutefois d'ajuster la réception du signal, la luminosité de l'image et le volume du son. Les utilisateurs pouvaient également se voir et régler la focale de la caméra. Enfin, le système disposait d'un mode "discret" limitant la transmission à la voix.

La bande passante nécessaire à une liaison entre deux Picturephones était si rare que AT&T avait du mal à en faire la démonstration. Le système fut pourtant présenté à l'occasion de l'exposition universelle de Flushing Meadow (New York) en 1964. Le public de l'exposition fut invité à essayer une unité qui était reliée à une unité identique située à Disneyland (Californie). Après avoir essayé le système, chaque utilisateur fut soigneusement interrogé par une agence de marketing dans le but de déterminer le potentiel commercial de ce type de produit. Le résultat de cette enquête ne fut pas celui attendu. Les gens n'aimaient pas le Picturephone. Ils le trouvaient trop volumineux, trop froid et se plaignaient de la taille trop réduite de l'image.

Les laboratoires Bell restèrent pourtant convaincus que le Picturephone était commercialement viable. Les essais continuèrent

pendant six années et fin 1970, un service interurbain basé sur un Picturephone de seconde génération fut commercialisé dans plusieurs grandes villes américaines telles que Pittsburgh, Chicago, New York ou Washington. Les cadres de AT&T étaient alors confiants : un million de Picturephone seraient vendus d'ici 1980.

Une scène du film *2001 l'odyssée de l'espace* [Kub68] illustre bien cette conviction que le visiophone allait révolutionner notre vie quotidienne. Dans cette scène, l'un des personnages utilise un Picturephone pour souhaiter un bon anniversaire à sa fille depuis une station en orbite autour de la Terre (Figure 1). La démonstration technique est impressionnante : la connexion espace/Terre est établie plus rapidement qu'une liaison téléphonique classique et n'est facturée que $1.7 pour plusieurs minutes de communication. Véritable publicité pour AT&T, cette scène ne manque pas de sens comique. Lorsque le père demande à sa fille ce qu'elle veut pour son anniversaire, celle-ci répond "un téléphone"...

Figure 1. Le Picturephone dans *2001 l'odyssée de l'espace*

Malgré les efforts considérables que AT&T lui consacra pendant plus de 15 ans, le Picturephone n'eut jamais de succès commercial. Après avoir dépensé plus de 500 millions de dollars de développements sur ce projet, la compagnie décida d'y mettre un terme à la fin des années 70, concluant que le visiophone était un "concept à la recherche d'un marché". Malgré ses améliorations successives, le Picturephone restait imposant, cher, et nécessitait toujours une importante bande passante difficilement disponible.

A la même époque, un premier réseau visiophonique était expérimenté à Paris par le CNET[2], puis étendu à Lannion par le réseau hertzien [BG96]. Ce réseau, utilisé jusqu'en 1980, desservit jusqu'à 200 postes installés sur les deux sites du CNET et au ministère des PTT. En 1984, le CNET expérimenta la visiophonie à l'échelle d'une ville. Pendant cinq ans, un réseau de fibres optiques permit à 1 400 habitants de Biarritz de communiquer par visiophone. Ce type d'expérience ne fut pas par la suite généralisé, officiellement pour des raisons de coûts. Dix ans après l'expérience

[2] Centre National d'Etude des Télécommunications, aujourd'hui France Télécom R&D

américaine, et malgré l'évolution des technologies utilisées, l'idée de visiophone fut une nouvelle fois abandonnée.

1.2. Salles de visioconférence

A la fin des années 70, constatant l'échec du visiophone, plusieurs sociétés américaines ont commercialisé des systèmes de communication audiovisuelle non plus à usage individuel mais destinés à des réunions de groupe. L'acquisition et la mise en œuvre de ces systèmes relevaient encore à cette époque de l'acte de foi : le codec[3] coûtait $250 000 à lui seul auquel il fallait ajouter des caméras, moniteurs, microphones, haut-parleurs et généralement l'aménagement d'une salle spécifique pour accueillir tout ce matériel (Figure 2). Un certain nombre de grandes entreprises tentèrent pourtant l'expérience, comptant sur la réduction des frais de déplacement de leurs employés pour financer leurs systèmes.

Figure 2. Salle aménagée pour la visioconférence

Ces salles de visioconférence s'avérèrent trop chères, trop complexes à utiliser et manquant de souplesse. Elles nécessitaient ainsi souvent la présence de techniciens et étaient difficilement adaptables au nombre de participants. Le terme *visioconférence* décrit bien le type d'usages auquel ces salles étaient généralement réservées : des situations regroupant un grand nombre de personnes qui assistent à une présentation formelle faite par un petit nombre d'entre eux.

L'idée d'utiliser la vidéo pour des *réunions*, et non plus des conférences, s'est imposée progressivement, accompagnant la miniaturisation et la standardisation des équipements. Les systèmes sont donc devenus plus simples, plus petits et mobiles.

1.3. Systèmes mobiles de téléréunion

Les systèmes mobiles de téléréunion regroupent codec, écran(s), caméra(s), haut-parleurs et microphone(s) sur un meuble qui peut être déplacé (Figure 3). Contrairement aux premiers systèmes de

[3] élément responsable du codage et décodage de l'audio et de la vidéo

visioconférence, ceux-ci n'utilisent pas de ligne spécialisée mais transmettent le son et l'image sur plusieurs lignes téléphoniques numériques ISDN - ou NUMERIS -, une à trois lignes suffisant à obtenir une qualité plus qu'acceptable.

Figure 3. Système mobile (Compression Labs Incorporated, 1997)

Les systèmes mobiles sont moins coûteux, plus simples et plus souples d'utilisation que les salles dédiées à la visioconférence. Ces avantages ont naturellement conduit à une augmentation de la demande et donc à une multiplication des offres. Cette multiplication des offres a rendu nécessaire l'interopérabilité entre les systèmes et a conduit à une standardisation des techniques de transmission. Le codage du son et de l'image, le contrôle multipoint, et, dans une certaine mesure, le partage de données ont été normalisés sous l'impulsion de l'Union Internationale des Télécommunications (ITU). Ces différentes normes[4] permettent aujourd'hui aux différents systèmes de communiquer entre eux indépendamment du réseau, de sa topologie, de la plate-forme ou de l'application utilisés.

Les systèmes modernes de communication vidéo de groupe sont mobiles... comme les ordinateurs transportables des années 80. Le poids de l'ensemble et le nombre de câbles limitent généralement les possibilités de déplacement de ces systèmes. En conséquence, nombre d'entre eux sont installés de façon permanente dans une salle qui ne leur est pas nécessairement dédiée. Malgré les autres avantages des systèmes mobiles sur les salles de visioconférence, leur utilisation passe donc souvent également par une réservation préalable, ce qui les limite une nouvelle fois à des usages formels et planifiés.

[4] Deux familles de normes ont été créée : la famille H320 pour la compression et transmission de l'audio et de la vidéo, et la famille T120 pour le transfert et partage de données. Pour plus de détails sur ces normes, consulter http://www.itu.org/

1.4. L'ordinateur personnel communiquant, ou le renouveau du visiophone

La miniaturisation des composants, l'extension des réseaux et l'augmentation de la puissance de calcul des ordinateurs ont accompagné la standardisation des protocoles de communication vidéo. La majorité des systèmes de téléréunion actuels reposent sur une machine de type PC équipée d'une ou plusieurs cartes pour l'acquisition, le codage et la transmission des données. Près de trente ans après le Picturephone, cette convergence des équipements audiovisuels et de l'ordinateur personnel offre de nouvelles perspectives pour la communication médiatisée. De nombreuses applications destinées au grand public sont ainsi aujourd'hui disponibles, permettant de transformer un "PC multimédia" en visiophone personnel (Figure 4).

Figure 4. CU-SeeMe (White Pine Software, 1997)

Les qualités de transmission et de restitution audio et vidéo de ces systèmes sont encore, pour le moment, inférieures à celles offertes par les systèmes de groupe. Cependant, l'intégration du système de communication à l'environnement informatique devrait à terme faciliter l'échange ou le partager de données. La conception de ces systèmes se heurte malheureusement souvent aux difficultés de conception d'interfaces collaboratives. La volonté de vouloir tout faire, de la communication audiovisuelle au transfert de fichier et au tableau blanc, ainsi qu'un usage excessif de la métaphore du téléphone conduisent encore trop souvent à une multiplication des interfaces qui rend ces systèmes difficilement utilisables (Figure 5).

Figure 5. ShareVision (Creative Technology, 1997)

Les progrès techniques réalisés depuis une vingtaine d'années pourraient offrir une seconde chance aux systèmes de communication vidéo à usage individuel. Mais les systèmes proposés jusqu'à présent se contentent d'adapter les solutions de groupe et de reproduire les usages classiques du téléphone. Nous allons voir dans les sections suivantes que les usages potentiels de la vidéo ne se limitent pas à ces situations formelles.

2. L'informel et le quotidien

Depuis le Picturephone, de nombreuses études se sont penchées sur l'apport de la vidéo pour la communication à distance. [PG92] et [TI93] soulignent ainsi l'aspect qualitatif de cet apport. La vidéo permet notamment d'accompagner la communication verbale de gestes, postures, mimiques qui permettent à leur tour d'interrompre une personne, d'attirer l'attention, d'exprimer un désaccord ou d'interpréter les silences.

D'autres études, telles que [Egi88] ou [MS94], se sont concentrées sur les raisons de l'échec des prédictions optimistes des premières années. Ces études montrent que le manque de succès des premiers systèmes n'était pas nécessairement lié à des raisons technologiques ou économiques mais reflétait surtout l'absence de prise en compte des facteurs psychologiques et sociologiques liés au déploiement et à l'utilisation de cette technologie. Deux expériences réalisées au cours des années 80 illustrent bien l'importance de ce contexte psychosocial en proposant précisément un usage des moyens techniques et un mode de développement inhabituels qui ont conduit à des usages informels : *Hole-in-Space* et *Media Space*.

2.1. Hole-in-Space

Un soir de novembre 1980, les artistes K. Galloway et S. Rabinowitz ont créé Hole-in-Space [GR80], la première "sculpture de communication publique". Ce soir là, les passants qui se trouvaient au *Lincoln Center for the Performing Arts* à New York, et au grand magasin *The Broadway* à Century City (Los Angeles) ont eu la surprise de se rencontrer.

Le dispositif comprenait à chaque site un grand écran muni d'une caméra et de haut-parleurs. Les connexions audio/vidéo étaient assurées par une liaison par satellite. De façon totalement imprévue, les passants se trouvèrent face à des images vidéo grandeur nature des personnes situées sur la côte opposée des Etats Unis (Figure 6). Ils pouvaient les voir, les entendre et parler avec eux comme si la rencontre avait lieu sur le trottoir d'à côté. L'installation ne comportait aucun panneau ou logo, aucune explication. Il n'y avait pas non plus de moniteur vidéo de contrôle afin de préserver l'impression de rencontre et de ne pas la transformer en visioconférence classique.

Figure 6. Hole-in-Space (*Electronic Cafe*)

Hole-in-Space supprima la notion de distance entre les deux villes pendant trois jours durant lesquels se succédèrent différents types d'événements. Il y eut d'abord le soir de la découverte, puis le soir du rendez-vous par le bouche à oreille et enfin une migration de masse où des familles, couples ou groupes séparés depuis parfois des années se retrouvèrent.

Cette expérience montre que la communication audio/vidéo peut être utilisée différemment des applications traditionnelles de type visiophone, visioconférence ou téléréunion. Qu'au-delà de la communication formelle, la même technologie permet de relier deux espaces et de créer un sentiment de téléprésence par une simple liaison permanente par l'image et le son entre deux espaces. Elle illustre également le fait que dans certaines situations, la meilleure interface soit l'absence d'interface explicite.

2.2. Media Space

Le terme "Media Space" a été inventé par R. Stults pour décrire un environnement audio, vidéo et informatique développé au milieu des années 80 [Stu86]. R. Stults faisait partie du *System Concepts Laboratory* de Xerox, un groupe de recherche destiné à explorer "l'informatique de groupe", suite logique du développement de l'informatique personnelle auquel le Xerox PARC avait déjà largement contribué.

L'un des thèmes qui intéressait plus particulièrement les chercheurs du SCL était la collaboration à distance. Afin de se placer dans des conditions réalistes pour étudier ce type de collaboration et développer les outils informatiques nécessaires, ils décidèrent de se répartir sur deux sites, l'un à Palo Alto (Californie) et l'autre à Portland (Oregon). Pour que les deux groupes distants puissent se réunir, R. Stults et ses collègues commencèrent par installer un système de visioconférence traditionnel dans des salles dédiées. Peu de temps après, ils décidèrent également de relier de façon permanente les espaces publics[5] des deux sites (Figure 7), recréant ainsi l'expérience Hole-in-Space.

[5] *The Commons*, l'espace de détente et de rencontre entre les différents bureaux

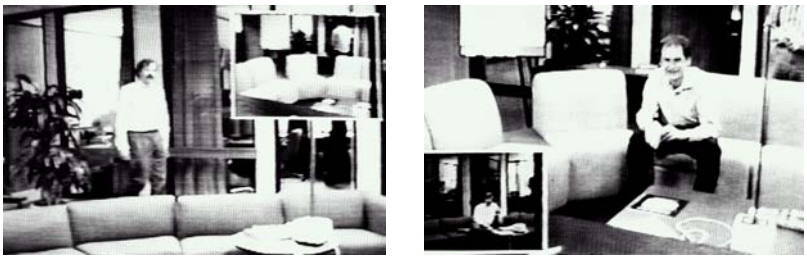


Figure 7. Espaces publics du Media Space (extrait de [BHI93])

Cette liaison permanente permettait aux personnes d'un site de rencontrer celles de l'autre site et de communiquer de façon informelle. Rapidement, plusieurs personnes voulurent bénéficier de ce type d'échanges avec leurs collègues depuis leur poste de travail. Elles équipèrent leur bureau en caméras, écrans, micros et haut-parleurs (Figure 8) et développèrent différents outils logiciels pour contrôler ces moyens audiovisuels par le système informatique afin de communiquer indifféremment avec les autres personnes, qu'elles soient sur le site distant ou dans le bureau d'à côté. Le Media Space fut ainsi utilisé pendant plusieurs mois de manière continue, à la fois comme outil et comme sujet de recherche. Lorsque les deux équipes furent regroupées à Palo Alto, il fut conservé et continua d'être utilisé malgré la proximité retrouvée.

Figure 8. L'un des bureaux du Media Space (extrait de [BHI93])

Comme Hole-in-Space, et à la différence des systèmes traditionnels de communication vidéo, le Media Space n'a pas été conçu dans un but précis, guidé par des impératifs technologiques ou commerciaux. Il est le résultat du travail d'un groupe de personnes aux motivations diverses dont le point commun était la volonté d'explorer de nouvelles manières de vivre en société et de collaborer. Au-delà de l'usage traditionnel pour des liaisons ponctuelles et formelles entre les deux sites, cette expérience montre que l'audio et la vidéo peuvent être intégrées dans l'environnement quotidien des utilisateurs pour leur offrir un accès permanent aux autres personnes, proches ou éloignées, favorisant ainsi les échanges informels et la coordination.

3. Partage de vues et fusion des espaces

Une bonne partie de notre activité quotidienne se fait sans utiliser de moyens informatiques. Même dans des environnements fortement informatisés, les utilisateurs passent sans cesse du monde réel au monde électronique, de leur bureau couvert de feuilles de papier à celui couvert d'icônes. Il nous arrive ainsi de devoir imprimer un document pour le relire et l'annoter, puis de reporter les corrections sur la version électronique.

De nombreux systèmes informatiques ont été développés pour la collaboration à distance. Ces systèmes permettent le partage et la manipulation de données informatiques mais sont généralement coupés du monde réel. La vidéo permet de relier non seulement des individus, mais également des espaces du monde réel. Il semble donc naturel de penser qu'elle pourrait être utilisée pour partager des objets du monde réel ou du moins leur image. Nous allons voir dans cette section que plusieurs travaux de recherche ont exploré ce nouvel usage de la vidéo, parmi lesquels *VideoDraw*, *TeamWorkStation* et *ClearBoard*.

3.1. VideoDraw

VideoDraw [TM91] est un système développé au Xerox PARC au début des années 90 qui permet à deux personnes distantes de partager une surface de dessin par l'utilisation de moyens vidéo. Sur chaque site, une caméra filme un large moniteur placé sous une vitre sur laquelle l'utilisateur local dessine avec des marqueurs effaçables. L'image ainsi capturée est alors envoyée vers le moniteur du site distant (Figure 9), des filtres polarisants placés devant écrans et caméras permettant d'éviter le phénomène de retour vidéo[6].

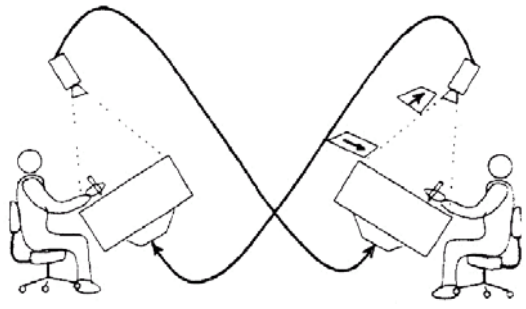


Figure 9. Configuration matérielle de VideoDraw (extrait de [TM91])

[6] Ce phénomène se produit quand des images filmées par une caméra et affichées par un moniteur entrent dans son champ de vision. Cette "récursivité" dans l'image finit en général par produire des taches lumineuse sur l'écran.

Lorsqu'une personne dessine sur l'écran, la caméra permet de transmettre les marques correspondantes mais également l'image de ses mains, et donc ses gestes, vers l'autre site. Chacun peut ajouter ses propres annotations au dessin de l'autre, et la combinaison de ces annotations faites en local avec l'image vidéo distante est à tout moment visible. Les utilisateurs peuvent dessiner, effacer, désigner sur l'écran de VideoDraw comme ils le feraient sur une feuille de papier. Ce système peut bien entendu être associé à une liaison audio pour faciliter davantage la communication entre les deux sites.

Quelques limites ont toutefois été observées à l'usage de VideoDraw. En raison de la faible résolution des caméras, les utilisateurs sont obligés de se servir de marqueurs épais. En conséquence, la surface de dessin offerte par les écrans 20 pouces se trouve rapidement entièrement couverte de marques, obligeant les utilisateurs à effacer celles jugées inutiles ou trop anciennes pour avoir de la place.

Le fait que les marques des participants soient sur des surfaces différentes, bien que visibles simultanément, pose également quelques problèmes. Un utilisateur ne peut pas effacer ou modifier lui-même les marques effectuées sur l'autre site. L'épaisseur de la vitre introduit de plus un léger décalage, ou effet de parallaxe, entre les marques locales et celles du site distant qui rend difficile les tracés de précision. Les marques locales et distantes apparaissent également avec des résolutions et intensités différentes. Ces différences peuvent être gênantes pour percevoir la vue composée comme une seule image. Dans certains cas, au contraire, elles peuvent s'avérer utiles : deux architectes discutant d'un bâtiment peuvent ainsi les utiliser pour représenter chacun un niveau du plan.

3.2. TeamWorkStation

VideoDraw n'utilise que de l'équipement vidéo et ne permet donc de capturer les gestes et les marques que de façon temporaire. Différents prototypes de recherche ont cherché à étendre ce type de système afin de pouvoir manipuler des objets du monde réel, comme des feuilles de papier, et de bénéficier en même temps des fonctionnalités classiques d'un système informatique, telles que la sauvegarde ou l'impression.

TeamWorkStation [IM91] est l'un de ces prototypes, développé par les laboratoires d'Interaction Homme-Machine de NTT[7]. TeamWorkStation combine audio, vidéo et informatique pour fusionner plusieurs espaces de travail en les superposant par transparence. La superposition est effectuée par le système

[7] Nippon Telegraph and Telephone

informatique, et non plus par des moyens optiques, ce qui permet de combiner des images du monde réel, une personne travaillant sur une feuille de papier par exemple, avec des images du monde informatique, comme le plan ou une vue en trois dimensions d'un bâtiment.

Figure 10. Configuration matérielle de TeamWorkStation (extrait de [Ish99])

Le système utilise deux caméras, l'une pour filmer la personne et l'autre pour filmer la surface de travail devant elle (Figure 10). Un écran informatique est utilisé pour afficher la superposition des différents espaces locaux et distants, informatiques et réels (Figure 11). Il est ainsi possible de désigner ou d'annoter avec la main et un feutre ou d'utiliser la souris et un logiciel de dessin. Les images des participants peuvent également être ajoutées à la composition vidéo.

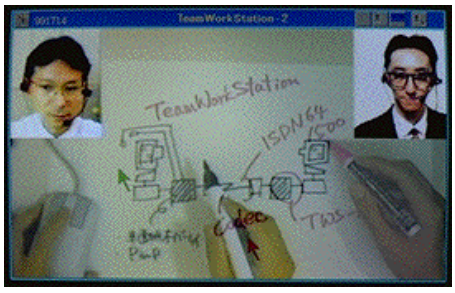


Figure 11. Exemple d'utilisation de TeamWorkStation (H. Ishii)

TeamWorkStation offre une plus grande flexibilité que VideoDraw. Cependant, bien que les différentes vues soient superposées et visibles comme un unique espace, les marques effectuées hors du système informatique ne sont toujours produites que sur un seul site. Comme avec un système purement optique, lorsque la session est terminée, chacun dispose seulement d'une partie des annotations. La seule façon de sauvegarder la vue globale est de capturer une image ou de fabriquer une séquence vidéo. La liaison vidéo supplémentaire montrant les deux participants leur offre de nouveaux moyens de coordination, mais la disposition de ces

images et la séparation entre l'écran informatique et les objets physiques manipulés rendent le contact visuel difficile.

3.3. ClearBoard

ClearBoard [IKG93] a été développé par l'équipe de TeamWorkStation afin d'explorer les possibilités de capture des annotations par le système informatique et de permettre une interaction naturelle autorisant la communication par gestes ou mouvements de tête et le contact visuel entre deux utilisateurs. ClearBoard repose sur une métaphore simple : les utilisateurs doivent avoir l'impression de se voir à travers une glace sur laquelle ils peuvent dessiner (Figure 12).

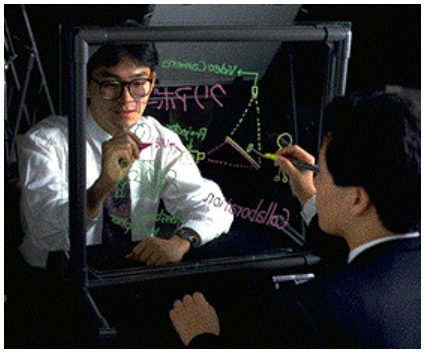


Figure 12. La métaphore de ClearBoard (H. Ishii)

Sur chaque site sont disposés une caméra, un projecteur et un film transparent de numérisation placé sur un écran de 80 x 60 cm (Figure 13). L'écran permet de voir à la fois l'image de l'utilisateur distant ainsi qu'une application de dessin partagée. Le film de numérisation permet d'utiliser cette application en traçant directement sur l'écran.

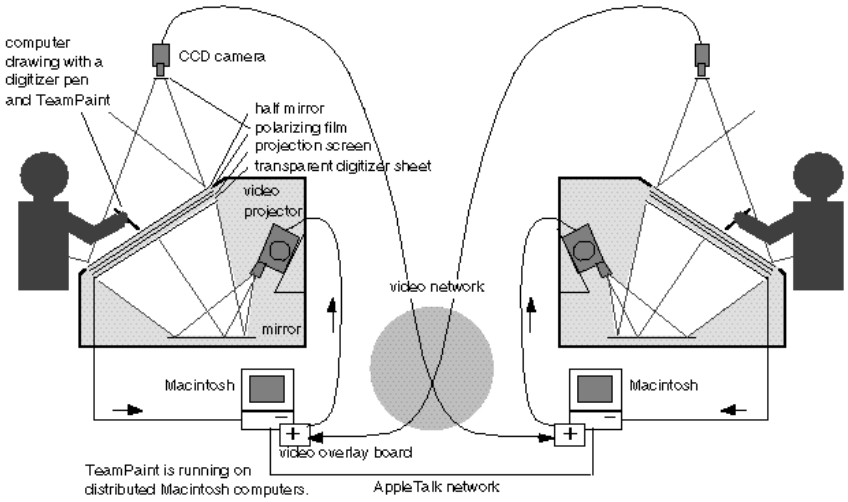


Figure 13. Configuration matérielle de ClearBoard (extrait de [IKG93])

La Figure 14 montre ClearBoard en situation d'utilisation. L'image de l'utilisateur distant est inversée pour conserver l'orientation gauche/droite relative à la surface de dessin. Les utilisateurs peuvent donc lire le texte et les dessins correctement. Le contact visuel est également possible : l'utilisateur peut déterminer le centre d'intérêt de la personne distante en suivant son regard et sait si cette personne le regarde. L'application partagée permet d'éditer un dessin puis de le sauvegarder en vue d'un usage ultérieur ou d'importer des documents provenant d'autres applications.

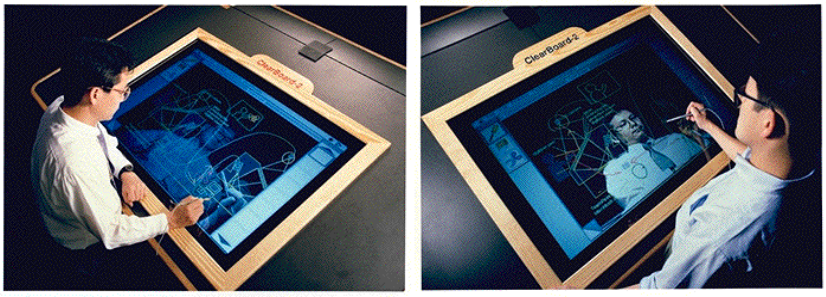


Figure 14. ClearBoard (extrait de [IKG93])

Bien que limité à deux participants, ClearBoard marque une étape importante dans l'usage de la vidéo pour la collaboration à distance. Pour la première fois, contrairement aux systèmes de communication vidéo classiques, un prototype a su intégrer de façon naturelle la communication vidéo avec l'outil informatique pour offrir un réel espace de collaboration partagé. Nous allons voir dans la section suivante une dernière catégorie d'usages de la vidéo qui repose sur le traitement des images.

4. Traitement d'images pour l'interaction homme-homme ou homme-machine

Tous les usages de la vidéo que nous avons vus jusqu'ici sont basés sur une communication entre plusieurs personnes par la reproduction à l'identique d'images capturées par des caméras. Le rôle croissant de l'informatique dans l'acquisition, la transmission et la restitution des images a cependant ouvert de nouvelles perspectives liées à la possibilité de traitement en temps réel de ces images. La machine étant capable de transformer ou d'analyser les images, il est par exemple possible de projeter l'utilisateur dans un monde virtuel ou de le suivre et d'interpréter ses gestes. Nous présentons ici ces deux approches, que nous désignons par les termes de *Réalité artificielle* et de *Réalité augmentée*.

4.1. Réalité artificielle

Le concept de réalité artificielle a été inventé à la fin des années 60 par M. Krueger. Une expérience de réalité artificielle est une forme avancée de simulation informatique, qui peut être envisagée dans

un but pratique, mais qui peut également être considérée comme une forme d'art interactif. En y participant, les utilisateurs ou visiteurs entrent dans un monde de synthèse où ils peuvent interagir avec d'autres personnes distantes ou simplement s'amuser en jouant avec des créatures graphiques. La clef du succès d'une expérience de réalité artificielle réside dans la réponse aux besoins, désirs et actions des participants.

Dès 1969, M. Krueger crée une série d'environnements interactifs qui mettent l'accent sur une participation multi-sensorielle et une grande liberté d'expression. Influencé par une culture artistique libérale, il accorde une importance philosophique à la rencontre de l'homme et de la machine et s'intéresse plus particulièrement aux problèmes d'interaction. Selon lui, l'interface ultime doit être le corps humain et ses cinq sens. Comme il le décrit dans son livre, la réalité artificielle est une interface générique, un moyen de communication et d'expression entre la machine et l'homme [Kru91]. Il insiste sur le fait que cette interaction doit se faire sans aide extérieure, sans accessoire. C'est ce qui différencie son approche de la *Réalité virtuelle*, qui nécessite généralement le port de lunettes et de gants spéciaux pour plonger l'utilisateur dans le monde artificiel.

M. Krueger a créé *Videoplace*, un ensemble de technologies permettant d'expérimenter le concept de réalité artificielle. Videoplace perçoit les actions des participants en termes de relations entre leur corps et un monde simulé. Il produit ensuite des images et des sons qui créent l'illusion convaincante de la participation des êtres humains à ce monde simulé. La machine pouvant contrôler des moyens vidéo, de l'infographie et de la musique électronique, de riches relations peuvent être mises en place entre un individu et l'environnement. L'environnement peut être contrôlé par des programmes prédéfinis, mais il peut également servir à amplifier les possibilités d'interaction entre les individus.

Videoplace permet de mélanger des images du monde réel avec des éléments graphiques de synthèse. Un projecteur vidéo présente l'image composée sur un écran de grande taille placé face à l'utilisateur. Une plaque de plastique blanc placée derrière celui-ci et une caméra lui faisant face permettent à un système de vision par ordinateur de capturer la silhouette de l'utilisateur. Les silhouettes de plusieurs personnes peuvent ainsi être affichées simultanément, chaque image pouvant être positionnée et dimensionnée de façon arbitraire (Figure 15, à gauche). En plus des objets graphiques statiques et des images des participants, Videoplace peut introduire un élément graphique animé, baptisé *Critter*, dont le comportement est lié aux événements qui se déroulent dans l'espace virtuel. Critter interagit avec les participants de manière ludique. Par exemple, si un participant bouge, Critter le suit ; s'il tend sa main ouverte, il se pose dessus (Figure 15, à droite).

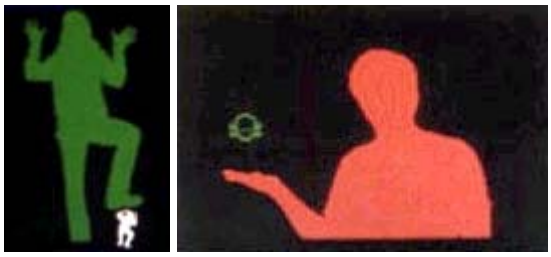


Figure 15. Videoplace (*Ars Electronica Center*)

M. Krueger a approché la création de mondes virtuels d'un point de vue esthétique, utilisant l'ordinateur pour créer des œuvres d'art qui permettent à plusieurs participants d'interagir en temps réel avec des images générées par la machine. Bien que les relations de causes à effets soient choisies par l'artiste, les mouvements des participants déterminent complètement ce qu'ils vont percevoir. Concernant ces mouvements, il est intéressant de noter que M. Krueger a pu observer de nombreux comportements de la vie réelle reproduits dans le monde virtuel. En particulier, les utilisateurs de Videoplace se déplaçaient instinctivement pour faire de la place aux autres personnes dans l'image composée [Kru91].

L'analyse d'image permet à Videoplace de détecter le contact entre l'image d'une personne et celle d'un autre participant ou d'un objet graphique. Le système peut donc répondre à ce contact en déplaçant par exemple l'objet ou en le faisant disparaître. Les techniques de vision utilisées permettent également aux participants de dessiner avec leurs doigts ou avec leur corps tout entier (Figure 16).

Figure 16. Utilisation du geste dans Videoplace (*Ars Electronica Center*)

En 1987, M. Krueger et son équipe ont poursuivi dans cette voie en développant *Videodesk*, un prototype destinée à exploiter le langage gestuel humain dans les interfaces homme-machine. Videodesk utilise une tablette lumineuse et une caméra placée au-dessus de cette tablette qui filme les mains de l'utilisateur afin de capturer ses gestes. La silhouette des mains est superposée à l'affichage informatique et permet à l'utilisateur de dessiner ou d'écrire avec ses doigts mais également de désigner des objets. La main est alors utilisée comme une souris, l'utilisateur pouvant même cliquer en touchant son pouce avec son index.

4.2. Réalité augmentée, environnements réactifs et interfaces perceptuelles

Afin de réaliser ses différentes expériences de réalité artificielle, M. Krueger utilisait des circuits intégrés développés spécifiquement pour la segmentation du profil des participants. L'augmentation de la puissance de calcul des machines depuis les années 70 est telle que ces traitements peuvent aujourd'hui être effectués par n'importe quel ordinateur personnel. Les machines actuelles sont capables d'analyser en temps réel un flux vidéo pour en extraire diverses informations permettant de localiser, d'identifier ou de suivre des objets présents à l'image. La vidéo ne se limite donc plus à l'affichage d'images, mais peut être également utilisée comme périphérique d'entrée pour l'interaction homme-machine.

Depuis les premières expériences de M. Krueger, d'autres travaux ont combiné des techniques de projection et de traitement de flux vidéo pour fusionner l'environnement informatique et l'environnement réel. Le *bureau digital* (*Digital Desk* [Wel93]) imaginé par P. Wellner permet par exemple de sélectionner des chiffres destinés à un calcul en les désignant sur une feuille de papier (Figure 17). W. Mackay et al. ont désigné sous le terme de *Réalité augmentée* cette nouvelle approche qui permet d'interagir avec le système informatique en manipulant des objets familiers comme le papier et les crayons [MVC+93]. [Mac96] et [Azu97] donnent un aperçu des nombreux autres travaux qui ont suivi cette voie, couvrant des domaines d'applications qui vont de la maintenance de photocopieurs et d'avions à la conception architecturale ou au diagnostic médical.

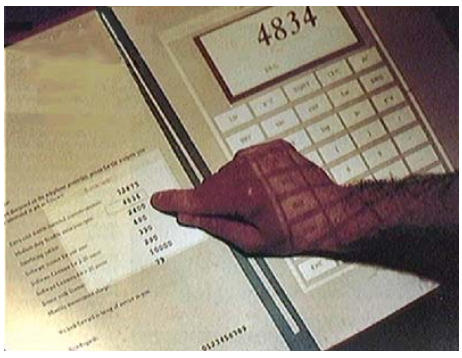


Figure 17. Digital Desk (extrait de [Wel93])

Les techniques de traitement d'images s'intègrent naturellement bien dans un système de communication vidéo où les flux d'images sont déjà présents. W. Gaver et al. ont ainsi conçu la *fenêtre virtuelle* [GSO95], un dispositif destiné à des situations de communication vidéo entre deux sites qui détecte les mouvements de l'utilisateur local et commande en conséquence le mouvement de la caméra distante. Les déplacements de l'utilisateur détectés grâce à la caméra locale lui permettent ainsi de changer de point de vue simplement en se déplaçant (Figure 18).

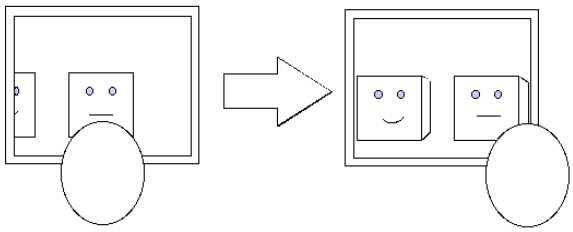


Figure 18. Principe de la fenêtre virtuelle

D'autres systèmes étendent dans un même esprit les possibilités offertes par un outil de communication vidéo existant. Le traitement d'images a ainsi été utilisé dans plusieurs systèmes de type Media Space pour renseigner les utilisateurs sur la présence récente d'un individu dans son bureau [LGS97, Bux97], et pour filtrer ou transformer les images transmises afin de préserver la vie privée des personnes présentes à l'image [HS96, CBCC98].

J. Cooperstock et al. ont introduit la notion d'*environnements réactifs*. La *salle réactive* (*Reactive Room* [CFBS97]) est une salle de visioconférence équipé d'un système informatique qui commande ses différents équipements (caméra, magnétoscope, éclairage, etc.) en fonction de l'activité des personnes présentes détectée par des techniques simples d'analyse des images et du son captés par les caméras et microphones. Ainsi, lorsqu'une personne place un document sous une caméra, le système de visioconférence bascule automatiquement sur cette caméra pour permettre aux personnes distantes de voir le document.

D'autres travaux continuent d'explorer les diverses possibilités de perception de l'homme par la machine, notamment à travers les images. Ces travaux font converger de multiples domaines de recherche, comme l'Interaction Homme-Machine, la Vision par Ordinateur, le Traitement du Signal, ou l'Intelligence Artificielle vers des interfaces qualifiées de *perceptuelles* [TR00]. [Bér99] démontre ainsi la faisabilité technique de dispositifs d'interaction fortement couplée entre l'homme et la machine à travers deux prototypes basés sur la vision par ordinateur : le *tableau magique*, un tableau blanc augmenté à la manière du bureau digital de P. Wellner et la *fenêtre perceptuelle*, un système qui permet à un utilisateur de contrôler le défilement du contenu d'une fenêtre sur un écran informatique par des mouvements de la tête.

La réalité augmentée, les environnements réactifs et les interfaces perceptuelles proposent le même type d'usage de la vidéo : celle-ci est considérée comme un nouveau moyen d'interaction entre l'homme et la machine, au même titre que les périphériques habituels tels que le clavier ou la souris. Comme nous l'avons vu, cette approche s'applique particulièrement bien à un contexte de communication médiatisée où elle bénéficie du matériel vidéo déjà utilisé et permet d'étendre les services existants, par l'ajout de

filtres de publication d'images par exemple, ou d'en proposer de nouveaux comme la fenêtre virtuelle.

5. Résumé du chapitre

L'échec des premières expériences de visiophone explique l'orientation des systèmes commerciaux vers des usages de groupe et plus particulièrement vers des situations formelles comme les conférences et les réunions. Les différents prototypes que nous avons présentés dans ce chapitre illustrent la diversité des usages de la vidéo que les progrès techniques récents permettent d'envisager. Au-delà des situations formelles et des usages de groupe sur lesquels se concentrent les systèmes commerciaux, nous avons vu que la vidéo peut être utilisée dans un contexte plus personnel pour l'interaction homme-homme ou homme-machine.

Hole-in-Space et Media Space montre que la vidéo peut être utilisée pour des formes de communication spontanées et informelles. VideoDraw, TeamWorkStation et ClearBoard montrent que la composition de flux vidéo peut être utilisée pour fusionner plusieurs environnements réels ou informatiques, permettant ainsi aux utilisateurs du système de se voir tout en partageant des objets physiques ou virtuels. En analysant et en transformant les images de ces flux vidéo réels et informatiques, les expériences de réalité artificielle de M. Krueger vont encore plus loin, changeant le système de communication vidéo en une œuvre d'art interactive. Les approches de type réalité augmentée illustrent plus encore l'intérêt de l'intégration des systèmes de communication audiovisuelle à des moyens informatiques de traitement des images, celles-ci pouvant alors être également utilisée pour l'interaction homme-machine.

Ce premier chapitre donne un aperçu des usages que nous voulons couvrir dans le cadre de notre travail sur l'intégration de la vidéo dans les systèmes informatiques interactifs. L'utilisation quotidienne et permanente du Media Space par ses concepteurs est particulièrement intéressante dans le cadre de ce travail. Nous allons voir dans le chapitre suivant qu'elle est en effet précisément liée à l'intégration du Media Space dans les habitudes de travail de ses utilisateurs.

Chapitre II

Mediaspaces

Le chapitre précédent nous a permis de présenter un certain nombre de systèmes utilisant la vidéo pour permettre la coordination, la communication ou la collaboration entre plusieurs personnes. La plupart de ces systèmes ont été développés dans un but précis, spécifique ou ponctuel comme le partage d'une surface de dessin ou l'installation temporaire d'une œuvre d'art. Le Media Space, en revanche, se présente comme approche globale de la communication médiatisée qui ne préjuge pas de la nature de l'activité des utilisateurs mais leur propose un environnement audio, vidéo et informatique permettant d'augmenter l'espace physique.

Dans ce chapitre, nous revenons sur ce concept en évoquant en détail le projet initial ainsi que ceux qui l'ont suivi, en étudiant leur spécificité et en présentant trois dimensions qui sont, selon nous, caractéristiques de ces environnements.

1. Du Media Space aux mediaspaces

Un mediaspace n'est pas un système de communication comme les autres. Ce n'est pas une nouvelle application, mais un ensemble de technologies qui augmentent l'espace physique[8] et offrent ainsi de nouvelles possibilités de coordination, de communication et de collaboration. Notre définition du mediaspace est la suivante :

[8] En termes anglo-saxons, nous parlerions d'*enabling technologies*

> Un *mediaspace* est un environnement matériel et logiciel qui associe des moyens audio, vidéo et informatiques pour assister des groupes de personnes dans leurs activités quotidiennes.

Depuis bientôt 20 ans, de nombreux projets de recherche ont exploré les problèmes techniques et sociaux posés par les mediaspaces [BHI93, SC93, Mac99]. Parmi les expériences les plus significatives, on peut citer :

- *VideoWindow* et *Cruiser* [CFKL92, FKRR92], développés par BellCore à la même époque que le Media Space et qui sont très proches de celui-ci ;
- *RAVE* [GMM+92] et *Kasmer*, de Xerox, descendants directs du Media Space et à l'origine de la plus grande partie de la littérature sur les mediaspaces ;
- *CAVECAT* [MBS+91] de l'Université de Toronto, cousin de RAVE développé à partir de celui-ci dans le cadre de l'*Ontario Telepresence Project*. Ce projet reste à ce jour la plus grande expérience réalisée au niveau des moyens mis en œuvre (trois ans et 5 millions de dollars US) ;
- *Montage* [TR94] de Sun, l'un des premiers systèmes à utiliser de la vidéo numérique pour des formes de communication informelles ;
- *Argo* [GKMD94] de DEC, similaire à Montage mais qui en plus de liaisons audio/vidéo numériques, permet de partager des applications.

1.1. Infrastructure matérielle

Sur le plan technique, un mediaspace consiste en un ensemble de bureaux ou d'espaces publics reliés entre eux par des réseaux audio, vidéo et informatique. Les premiers mediaspaces utilisaient un réseau audio/vidéo analogique [CFKL92, BM90, GMM+92, MBS+91] : des équipements audiovisuels grand public ou semi-professionels connectés à une matrice de commutation, celle-ci étant pilotée par le système informatique (Figure 19).

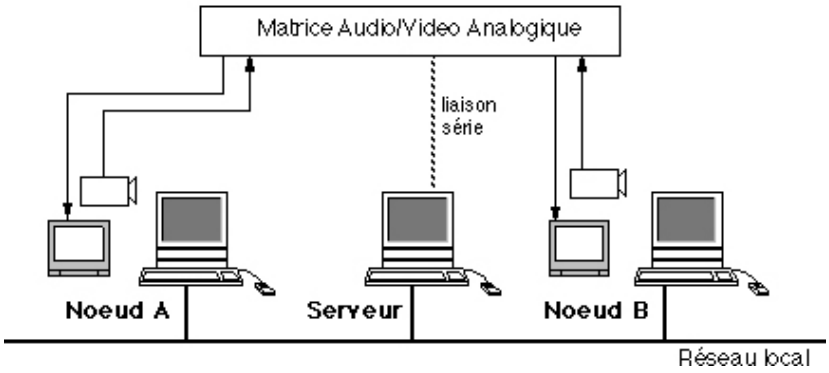


Figure 19. Configuration typique d'un mediaspace analogique. Les équipements audio sont omis dans un souci de simplicité

Un "nœud" typique de mediaspace analogique comporte une caméra vidéo avec microphone, un moniteur avec haut-parleurs et un poste de travail informatique qui permet d'utiliser le logiciel de gestion des connexions en plus des applications informatiques habituelles (Figure 20).

Figure 20. Configuration typique d'un nœud de mediaspace analogique.

Le choix d'une infrastructure analogique pour les premiers mediaspaces était principalement motivé par l'impossibilité d'obtenir une qualité équivalente avec les systèmes informatiques de visioconférence de l'époque, ceux-ci se préoccupant plus de la bande passante utilisée que de la qualité de l'image et du son restitués [Lyl93]. Cette qualité est particulièrement importante dès lors que l'on veut se concentrer sur les usages du son et de l'image et non sur la technique nécessaire pour les transmettre.

L'infrastructure analogique permettait également d'ajouter au réseau audio/vidéo des équipements peu coûteux pour filmer de façon continue des espaces publics comme les espaces de détente des laboratoires de Xerox ou le campus de l'Université de Toronto.

Par la suite, d'autres systèmes comme Argo ou Montage ont pu bénéficier des progrès réalisés en matière de réseaux et de

puissance de calcul des machines pour utiliser les réseaux informatiques pour transmettre les données audiovisuelles.

1.2. Services

Le développement des services des mediaspaces s'est fait au fur et à mesure des besoins exprimés par leurs utilisateurs. En résumant de manière synthétique les différentes expériences, nous pouvons décrire le scénario de l'évolution chronologique d'un mediaspace hypothétique :

1. Les lieux de communication formelle, tels que les salles de réunions, sont reliés par un système de visioconférence classique [BHI93].

2. Les zones de passage et de rencontre sont reliées, ce qui favorise les rencontres, la communication informelle et facilite la coordination entre les membres du groupe [BHI93].

3. Le réseau audio/vidéo est étendu à un ensemble de bureaux. Un service de type visiophone permet alors de choisir un correspondant ou une source publique et une option miroir permet de contrôler la prise d'image et de son locale [BHI93].

4. Un nouveau service de coordination est ajouté : le *coup d'œil*, une connexion de quelques secondes uniquement qui permet de savoir si quelqu'un et là sans trop le déranger. Afin de réduire encore le côté intrusif, la connexion peut éventuellement être limitée à la vidéo seulement et être unidirectionnelle. En exécutant une série de coups d'œil sur différents bureaux, les utilisateurs retrouvent les conditions rencontrées lorsqu'ils marchent dans un couloir en regardant au passage à travers les portes entrouvertes [CFKL92].

5. De nouvelles sources sont ajoutées au système : tuner T.V., magnétoscopes, caméras filmant l'extérieur des bâtiments. C'est l'occasion d'ajouter un service de connexion par défaut, interruptible et rétablie lorsque cette interruption est terminée [GMM+92]. Ce service permet d'être connecté de façon permanente à une personne, à une chaîne d'information en continu ou à une vue sur l'extérieur du bâtiment tout en restant accessible.

6. Conséquence logique de la possibilité de connexion par défaut, toujours active, certaines personnes "partagent" leurs bureaux pendant des jours, des semaines ou des mois [AH94, DABH96].

7. Les utilisateurs se tendent des embuscades [FKRR92] : ils restent connectés à un bureau vide pour être sur de savoir quand la personne sera de retour. Afin de pouvoir voir d'un seul coup d'œil qui est présent ou absent et avoir une impression globale de l'activité du groupe, un service de *vue d'ensemble*[9] est créé (Figure 21) : une application qui affiche une série de petites images fixes provenant des différents nœuds du mediaspace, capturées à intervalles réguliers [BT91, CFKL92, DB92, LGS97].

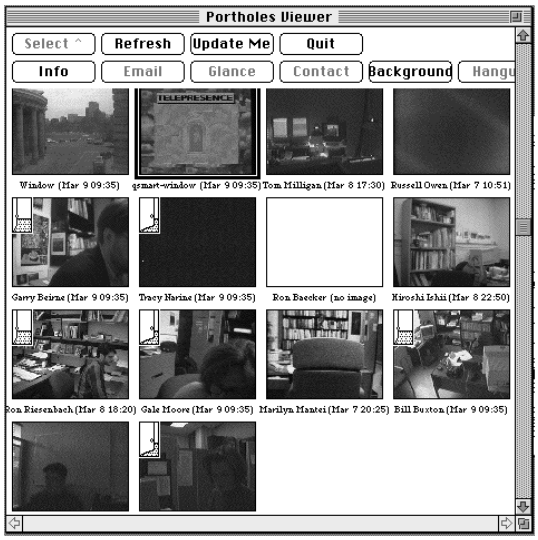


Figure 21. Vue d'ensemble de CAVECAT (Ontario Telepresence Project)

8. Afin de permettre les discussions à plusieurs, chacun restant dans son bureau, des dispositifs de mixage audio et vidéo sont ajoutés et un service de téléréunion est créé [BHI93]. Un environnement collecticiel permet d'associer des applications partagées à la communication [MBS+91, TR94, GKMD94].

L'accès aux services du mediaspace se fait généralement par un logiciel spécifique, par des choix dans les listes de services et d'utilisateurs. L'interface de RAVE est particulièrement intéressante. Elle se présente sous formes de boutons affichés en permanence sur l'écran informatique [BM90, MCLM90, GMM+92] (Figure 22). Chaque bouton correspond à quelques lignes de commandes qui déclenchent l'exécution d'un service. Les boutons peuvent être déplacés, copiés ou encore envoyés par courrier électronique. Le code qui leur est associé peut également être modifié par l'utilisateur. [GMM+92] décrit en détail comment ces possibilités de modification de l'interface du mediaspace ont

[9] C'est ainsi que nous traduirons l'expression *awareness view*

contribué à l'émergence et à l'évolution des services que nous avons mentionnés précédemment.

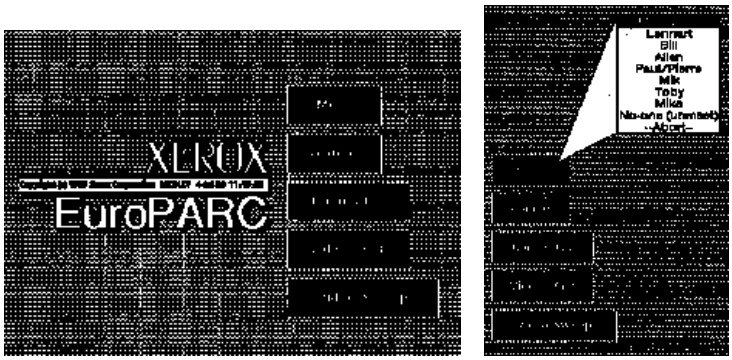


Figure 22. Interface de RAVE (extrait de [BM90])

1.3. Contrôle d'accès et notification

La liaison des espaces individuels par le mediaspace soulève de nombreux problèmes liés au contrôle de l'accès aux équipements audiovisuels. Très rapidement, les utilisateurs du mediaspace prennent conscience du danger que ces équipements peuvent représenter pour leur intimité et celle de leurs collègues.

Lorsque le système est introduit dans une petite communauté homogène, cette exposition de la vie privée peut être acceptée par les membres de la communauté. Ainsi, le premier Media Space n'offrait aucune possibilité de contrôle sur les connexions. Toutes les demandes aboutissaient et la seule protection offerte était la possibilité de débrancher les caméras et les micros. Cette solution, bien qu'efficace, est également radicale et était rarement utilisée. Pour l'appelant, il est en effet difficile de savoir si l'absence d'image ou de son est du à une panne ou à une volonté de l'autre personne, et lorsque l'on met au point un système géographiquement distribué, il est important de savoir localiser les pannes...

RAVE et CAVECAT ont été déployés dans des contextes sociaux plus complexes où se côtoyaient chercheurs et personnels administratifs. Dans ce type de contexte où la population est divisée en multiples sous-groupes, les utilisateurs souhaitent pouvoir contrôler plus finement l'accès à leurs moyens audiovisuels. Divers mécanismes de contrôle ont donc été mis en œuvre pour concilier accessibilité et vie privée.

La première méthode de contrôle utilisée dans les mediaspaces est la négociation explicite entre le système et la personne appelée, selon le principe du téléphone : "acceptez-vous l'appel de X ?". Si ce modèle offre un moyen simple pour contrôler les demandes de connexion, il crée également de nouveaux problèmes : comment savoir si une personne est là sans la déranger ? Comment refuser une communication sans vexer celui qui l'a demandée ? Ainsi, face à cette dernière question, les utilisateurs de Cruiser préféraient ne

pas répondre à un appel en laissant croire qu'ils étaient absents, plutôt que de devoir exprimer clairement le refus [CFKL92]. Ce type de réaction est le plus souvent lié au manque d'information sur l'identité de l'appelant et son intention. L'espace physique qui nous entoure nous fournit de nombreux indices visuels et auditifs qui permettent de juger de la disponibilité des autres personnes sans pour autant s'engager dans une communication directe. Ainsi, un simple coup d'œil à travers une porte ne perturbe pas la personne aperçue mais renseigne néanmoins sur son activité et sa disponibilité.

Les concepteurs et les utilisateurs de RAVE ont choisi de différencier le partage de bureau et le visiophone. Ces deux services sont pourtant identiques en termes d'utilisation des ressources audiovisuelles, puisqu'ils utilisent tous deux une liaison bidirectionnelle audio et vidéo. Le partage de bureau et le visiophone se différencient non pas par des critères techniques, mais par des critères d'usage : la durée de la communication et le type d'échanges attendu. La diversification progressive des services telle que nous l'avons décrite correspond à la reconnaissance de la diversité des usages. En nommant les usages observés et en les proposant comme services, les concepteurs du mediaspace offrent aux utilisateurs un moyen d'exprimer l'intention liée à une demande de connexion. Associée à l'identité de l'appelant, cette expression de l'intention donne à la personne appelée une indication du degré d'engagement attendu [GMM+92] (Figure 23) qui l'aide à décider de la suite à donner à la demande.

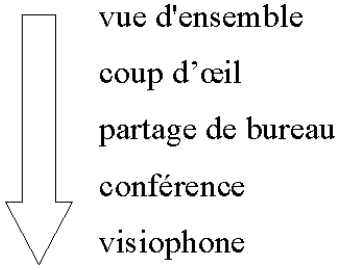


Figure 23. Classement des services par degré croissant d'engagement attendu

Dans Cruiser, comme dans le premier Media Space, toutes les connexions étaient réciproques, c'est-à-dire bidirectionnelles. La diversification des services a rapidement montré que cette réciprocité stricte est trop intrusive [CFKL92]. Dans le cas du coup d'œil par exemple, s'il est légitime que la personne contactée soit avertie que quelqu'un la regarde, l'apparition soudaine sur son écran de tous ceux qui jettent un coup d'œil peut s'avérer gênante. Divers mécanismes de notification ont donc été mis en œuvre afin de limiter le côté intrusif du mediaspace tout en maintenant les utilisateurs informés de l'activité du système.

Les notifications sont un ensemble de commandes déclenchées avant, pendant ou après l'exécution d'un service pour indiquer aux utilisateurs du mediaspace l'activité du système. Ces notifications peuvent prendre différentes formes : boîtes de dialogues, courrier électronique, traces dans un fichier, sons échantillonnés ou synthétisés, etc. L'utilisation du son se révèle particulièrement intéressante puisqu'il permet d'informer efficacement l'utilisateur sans monopoliser son attention [Con97]. Ainsi, RAVE permet d'associer des sons usuels échantillonnés aux différents services, tel que le bruit d'une porte qui grince pour les coups d'œil ou un déclenchement d'appareil photo lorsqu'une image est numérisée pour la vue d'ensemble.

La notification peut combiner des aspects visuels et auditifs pour recréer l'approche non intrusive du monde réel. Montage utilise par exemple un fondu d'images et une montée progressive du niveau sonore pour établir les connexions, évitant ainsi l'apparition soudaine d'une personne sur l'écran. La Figure 24 montre l'utilisation de cette approche progressive pour annoncer un coup d'œil. Le coup d'œil n'établit par défaut qu'une liaison vidéo bidirectionnelle, mais une icône de haut-parleur et un bouton "Visit" permettent aux deux personnes connectées d'ajouter une liaison audio ou de le transformer en liaison de type visiophone, dont la durée n'est pas limitée et qui offre des images plus grandes.

Figure 24. Fondu d'images pour annoncer un coup d'œil (extrait de [TR94])

Associés à des mécanismes de contrôle d'accès, les notifications permettent de décrire des protocoles sociaux complexes tels que :

```
Accepter les demandes de visiophone venant des membres de mon
équipe. Pour les autres, une confirmation est nécessaire.

Pour les coups d'œil,
  si je ne suis pas déjà connecté, montrer la personne ;
  sinon, je veux entendre un bruit de porte
```

RAVE propose ainsi une série de dialogues qui permettent à l'utilisateur de spécifier pour chaque service la liste des personnes autorisées à l'exécuter ainsi que les actions à effectuer au début et à la fin de la connexion (Figure 25).

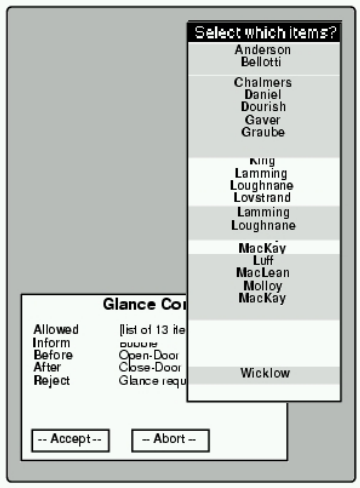


Figure 25. Panneau de contrôle de RAVE pour le contrôle et la notification liés au service de coup d'œil. L'utilisateur est en train de choisir la liste des personnes autorisées à demander ce service.

Plusieurs mediaspaces permettent également de choisir les règles à appliquer en indiquant le degré d'accessibilité voulu. Dans CAVECAT, les utilisateurs contrôlent ce degré d'accessibilité par une porte virtuelle (Figure 26).

Figure 26. Interface de CAVECAT
(Ontario Telepresence Project)

Selon l'état de cette porte, les demandes de connexion sont traitées différemment :

- si la porte est ouverte, toutes les demandes sont acceptées ;
- si elle est entrouverte, les coups d'œil sont autorisés mais les autres demandes nécessitent un accord explicite (il faut frapper avant d'entrer) ;

- si elle est fermée, l'accord explicite est nécessaire pour tous les services ;

- si elle est verrouillée, vous ne pouvez que laisser un message.

Dans le même esprit, Montage utilise également la métaphore de la porte et propose : *Accessible*, *Ne pas déranger*, *Fermé*, *Autre*. *Ne pas déranger* indique permet aux autres utilisateurs d'obtenir la connexion s'ils insistent, tandis que *Fermé* correspond réellement à une porte fermée. *Autre* permet à l'utilisateur de laisser un message expliquant pourquoi il refuse les connexions.

2. Spécificité des mediaspaces

Malgré quinze ans de présence dans les laboratoires de recherche, les mediaspaces n'ont pas dépassé le stade de prototype expérimental. La seule exception est *CorelVIDEO*, un logiciel commercialisé de 1995 à 1998 par la société canadienne Corel, partenaire industriel de l'Ontario Telepresence Project. Ce produit, à priori destiné aux entreprises, reprenait plusieurs idées développées dans le projet Telepresence, comme l'utilisation des vues d'ensembles, des notifications sonores ou du contrôle d'accès par la métaphore de la porte (Figure 27). En 1998 malheureusement, l'activité de communication vidéo de Corel a été cédée à une autre compagnie et ce produit a disparu.

Figure 27. CorelVideo (Corel Corporation, 1997)

Quelles sont les raisons qui expliquent que les mediaspaces n'aient pas eu plus de succès ? Quel peut donc être leur avenir ?

2.1. Les raisons de l'exception

Pourquoi les mediaspaces sont-ils restés au stade d'expériences de recherche ? La raison est bien évidemment à la fois technique et sociologique.

Comme nous l'avons mentionné précédemment, la plupart des mediaspaces développés ces quinze dernières années ont utilisé du matériel analogique pour la qualité d'image et de son qu'il offrait, sa facilité d'installation et son utilisation "gratuite" en termes de ressources informatiques. S'il est vrai que l'ajout d'un équipement au mediaspace ne nécessite le plus souvent que l'installation d'un nouveau câble, la gestion centralisée de l'ensemble des ressources devient vite un casse-tête lorsque les matériels s'empilent et que les câbles se croisent (Figure 28, gauche). La gestion efficace de ce réseau audio/vidéo nécessite d'importants investissements en temps, en matériel et en personnel qui peuvent aller jusqu'à la création d'un véritable studio semi-professionnel (Figure 28, droite). Cette importante infrastructure matérielle, bien qu'invisible aux utilisateurs, constitue un premier frein technologique important à l'évolution, au déplacement ou à la duplication d'un tel système.

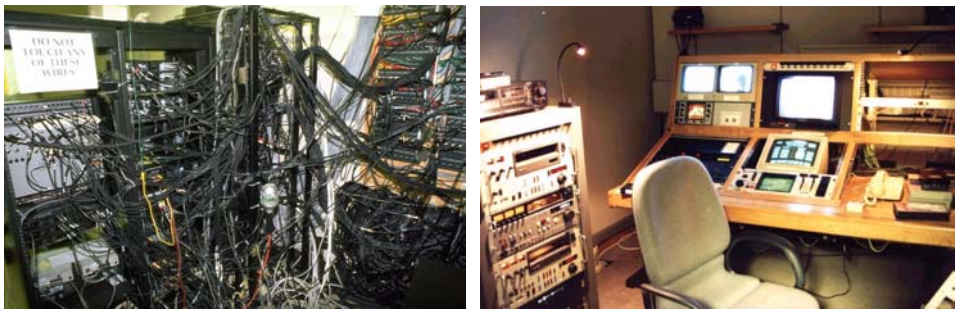


Figure 28. Une partie du câblage de CAVECAT (Ontario Telepresence Project) et l'*AV Suite* de RAVE (Xerox EuroPARC). Le panneau à côté des câbles indique : "*Do not touch any of these wires*"

Les logiciels utilisés pour commander les équipements analogiques ont été le plus souvent développés spécifiquement, en fonction des besoins et du matériel disponible. Le résultat de ces développements est le plus souvent un ensemble de programmes ad-hoc difficile à réutiliser. [BFK+94] rapporte ainsi que les chercheurs de l'Université de Toronto ont du totalement réécrire le logiciel de commutation de RAVE pour l'adapter au matériel de CAVECAT, puis que cette nouvelle version a du être réécrite une seconde fois lors de l'installation de leur propre système dans une entreprise. Ce manque de portabilité du logiciel existant constitue un deuxième frein technologique important à la mise en place d'un mediaspace.

Les problèmes sociaux posés par la mise en place d'un mediaspace peuvent être plus importants que les problèmes techniques. R. Fish va même plus loin : selon lui, les concepteurs de mediaspace n'ont quasiment aucun pouvoir sur l'utilisation effective du système et son acceptation [BFK+94]. Les problèmes de masse critique, de relations hiérarchiques entre les individus ou le délicat compromis entre accessibilité et vie privée sont bien plus déterminant pour le développement des usages que la technologie mise en œuvre.

La technologie seule ne peut en effet répondre aux problèmes sociaux causés par l'introduction de caméras et de micros. Ainsi, les connexions établies par le mediaspace ne constituent pas seulement un lien entre des équipements audiovisuels. Elles établissent également un lien entre les espaces [DABH96]. Ce partage de voisinage visuel et auditif, s'il n'est pas connu de toutes les personnes présentes, peut conduire à des situations pour le moins embarrassantes. En raison de l'absence de communication verbale directe, la plupart des personnes confrontées à un partage de bureau ne se rendent pas tout de suite compte que la liaison est à double sens. Des personnes de l'autre site placées en dehors du champ des caméras peuvent alors entendre et voir sans que les premières en aient conscience.

Nous voyons bien que ce type de problèmes ne peut être résolu que par l'adaptation du groupe au nouvel outil de communication. De nouveaux protocoles sociaux doivent être développés pour arriver à faire émerger une culture commune aux "habitants" du mediaspace [MBS+91, Dou93]. Le but des concepteurs est donc de fournir aux utilisateurs les éléments technologiques nécessaires à cette émergence. Ceci ne peut se faire que dans des contextes où les concepteurs sont suffisamment proches des utilisateurs pour pouvoir comprendre leurs attentes et proposer des alternatives permettant de résoudre les conflits. Cette démarche, proche de la conception participative [MK93], est très éloignée de la démarche habituelle de développement des produits de communication vidéo qui visent généralement le "grand public".

2.2. Quel avenir pour les mediaspaces ?

Nous venons de voir deux aspects des mediaspaces, l'un technologique et l'autre culturel, qui peuvent expliquer la difficulté de passage du stade expérimental au stade commercial ou grand public. Pour autant, les mediaspaces sont-ils des environnements réservés à quelques laboratoires de recherche à gros moyens ? Nous pensons que l'état actuel des possibilités techniques et des habitudes de communication sont favorables au développement d'une nouvelle génération de mediaspaces. Afin d'exposer notre point de vue, revenons un instant sur l'évolution des systèmes commerciaux de communication vidéo pour les comparer aux mediaspaces.

Depuis la commercialisation des premiers systèmes de visioconférence dans les années 60/70, le développement des technologies informatiques pour la communication poursuit le même rêve : recréer à distance les conditions d'une rencontre face à face [HS92]. Depuis 30 ans, l'idée qui dirige l'évolution des systèmes commerciaux est que la réalisation de ce rêve n’est qu’un problème d’utilisation optimale de la bande passante et de la puissance de calcul disponible. Les efforts de développement se

sont donc concentrés essentiellement sur la capture, la transmission et la restitution du son et de l'image.

Chaque nouvelle version des systèmes offre un son de meilleure qualité, des images plus grandes, de meilleures couleurs, une animation plus fluide et une latence plus faible. Cette progression des moyens techniques est régulièrement mise en avant par les services de marketing -"notre nouveau système va 50% plus vite"- qui promettent chaque année que le rêve est à portée de main -"bien entendu, plus il y a d'images par seconde, meilleure est la communication"- [Rie96]. Cette idée est tellement bien ancrée dans les mentalités que lorsqu'une expérience de communication vidéo échoue, cet échec est souvent attribué à l'insuffisance des moyens mis en œuvre - "ce n'est pas encore tout à fait au point, mais ce sera beaucoup mieux l'année prochaine" -.

Si les visionnaires des années 60 qui prévoyaient la fin des déplacements professionnels paraissent aujourd'hui ridicules, ce n'est pas parce que la chose nous parait infaisable, mais parce que nous sommes généralement convaincus qu'il est encore trop tôt. La conséquence de cet état d'esprit est la recherche perpétuelle du système qui aura enfin les capacités de traitement nécessaires à la téléprésence. Et dans cette perspective, pour convaincre les utilisateurs potentiels, il faut proposer plus.

Alors que la plupart des systèmes traditionnels se focalisent sur les situations de communication formelle, les mediaspaces ont montré qu'il est possible d'utiliser les moyens audiovisuels pour communiquer autrement, notamment de façon informelle. Plusieurs études ont montré l'importance de ce type de communication. Ces échanges brefs, fréquents, impromptus, sans réel début ou fin apparaissent comme essentiels pour la coordination et le développement de relations humaines mais disparaissent en l'absence de proximité physique [FKR93, IWFO97]. La spécificité du mediaspace est précisément cette possibilité de recréer les *effets* de la proximité physique, tels que la conscience de groupe et les opportunités de contact, et non la proximité elle-même.

De nombreuses observations sur l'utilisation des mediaspaces ont été effectuées dans des contextes où la co-présence des personnes étaient possible. [AH94] et [DABH96] décrivent ainsi des expériences de partage de bureau sur de très longues périodes, plus de deux ans, entre des personnes qui travaillaient dans le même bâtiment. Dans un tel contexte, il est évident que le but des utilisateurs du système n'est pas de remplacer le face à face qui est toujours possible, mais d'améliorer la communication au quotidien. Le rôle du concepteur de mediaspace est donc dans ce cas de proposer un système que les utilisateurs préfèrent à la proximité physique, de proposer des outils qui vont au-delà de la téléprésence [HS92].

Les travaux sur les mediaspaces montrent qu'au-delà des aspects quantitatifs liés à la transmission de l'image et du son, la communication est avant tout un phénomène social. Cette prise en compte du contexte de la communication peut avoir des conséquences surprenantes en termes quantitatifs. Dans certains cas, l'image peut par exemple être jugée trop grande, intimidante [Rie96]. Dans d'autres cas, les utilisateurs peuvent préférer une liaison sonore de moins bonne qualité mais qui leur permet aisément de distinguer les sons locaux des sons distants [Gav92]. Dès lors que l'on s'intéresse aux motivations des utilisateurs et au contexte de la communication, les valeurs ne sont plus celles auxquelles on était habitué : moins de données, ce peut être plus d'information. Et les services des mediaspaces qui paraissaient anecdotiques, comme les vues d'ensemble ou le partage de bureau, se révèlent beaucoup plus riches qu'on ne l'imaginait.

Comme nous l'avons fait remarquer au Chapitre I, les conditions technologiques pour la mise en place de systèmes de communication audiovisuels n'ont jamais été aussi bonnes. La puissance de calcul des machines actuelles permet de capturer, de transmettre et de restituer le son et l'image avec une qualité qui devient acceptable sans trop pénaliser les autres applications. L'explosion du phénomène Internet entraîne une multiplication des connexions à haut débit et une évolution rapide du matériel grand public qui permet d'avoir accès à cette puissance de traitement aussi bien chez soi que sur son lieu de travail pour un prix raisonnable.

Face à cela, le contexte social évolue rapidement lui aussi. La banalisation des téléphones portables et du courrier électronique tend à rapprocher les individus qui finissent par devenir dépendants de ces nouveaux outils, parfois malgré eux. La façon dont les gens gèrent leur courrier électronique est symptomatique de cette dépendance involontaire : de nombreuses personnes se sentent obligées de lire leur courrier lorsqu'ils sont en vacances pour ne pas devoir affronter une boîte aux lettres pleine en rentrant... La soif de communication que l'on observe aujourd'hui se heurte ainsi au seuil de frustration décrite par B. Buxton et évoqué dans le chapitre d'Introduction. Nous atteignons un point où les seules solutions proposées ne font qu'aggraver les choses : "trop d'appels sur votre téléphone portable ? prenez-en un deuxième ! ".

Nous pensons que nous sommes sur le point d'observer une transition importante dans l'évolution des technologies de communication. Les perspectives de développement sur le long terme ne peuvent plus s'exprimer en termes d'évolution quantitative des technologies actuelles. Il y aura bientôt assez de bande passante disponible pour transmettre en temps réel tout ce qui s'écrit ou s'enregistre. Nous arrivons au point où les utilisateurs ont besoin de gérer cette masse d'information. Ou la valeur d'un système se

mesure non plus par ce qu'il ajoute, mais par ce qu'il permet d'éviter. Dans cette perspective, les possibilités offertes par les mediaspaces pour la communication informelle et la coordination peuvent avantageusement compléter de nombreux outils de communication actuels comme le téléphone ou le courrier électronique.

3. Caractéristiques fondamentales des mediaspaces

Nous venons de voir que le contexte technique et social actuel semble favorable au développement d'une nouvelle génération de mediaspaces basés sur une infrastructure entièrement numérique. Ce passage de l'analogique au numérique devrait permettre de résoudre les problèmes de maintenance, d'extension et de reproductibilité des premiers systèmes. Mais qu'est-ce qui caractérise réellement un mediaspace ? Quels principes de base utiliser pour la conception des nouveaux environnements numériques ?

Dans cette section, nous revenons sur quelques points caractéristiques des premiers mediaspaces qui nous paraissent avoir joué un rôle primordial dans l'acceptation et l'utilisation de ces environnements par leurs utilisateurs. Nous identifions trois dimensions essentielles : l'*intégration* du mediaspace dans les habitudes des utilisateurs, sa *flexibilité* et le degré d'*intimité* qu'il assure. Nous montrons également comment ces propriétés peuvent être adaptées à la conception de l'infrastructure logicielle d'un mediaspace numérique.

3.1. Intégration

> L'*intégration* mesure le degré avec lequel un nouveau système s'insère dans les pratiques existantes des utilisateurs plutôt que d'en imposer de nouvelles

Comme le montre W. Gaver [Gav92], les propriétés physiques des éléments audio et vidéo peuvent favoriser ou au contraire limiter les possibilités de perception et d'interaction offertes par les mediaspaces. La disposition de ces éléments joue également un rôle fondamental dans l'intégration du mediaspace dans les habitudes de travail des utilisateurs [BHI93][10]. Sur ce point, les mediaspaces sont très différents des solutions traditionnelles de communication vidéo. Les salles de visioconférence imposent aux utilisateurs un changement de contexte explicite entre leurs activités personnelles et la collaboration avec des personnes distantes : ils doivent se

[10] Une fois de plus, le contexte d'utilisation est au moins aussi important que les caractéristiques de base…

déplacer dans un lieu spécifiquement dédié aux activités de groupe. Le nombre de salles étant généralement limité, les utilisateurs sont également obligés de planifier longtemps à l'avance les réunions. A l'opposé, les mediaspaces "augmentent" l'espace physique en intégrant les dispositifs de communication dans l'environnement personnel de chacun et en les rendant instantanément accessibles de façon permanente.

Une autre différence importante entre les mediaspaces et les systèmes traditionnels est la durée des communications. Comme nous l'avons déjà mentionné, certains utilisateurs de mediaspaces sont restés connectés pendant plusieurs années, ce qui leur a permis d'établir une véritable relation de partage de l'espace visuel et auditif [AH94, DABH96]. La robustesse du matériel et de la technique de commutation analogique a certainement contribué à rendre possible des connexions aussi longues. La possibilité de disposer d'un écran indépendant dédié au mediaspace a également permis d'isoler la communication des autres tâches pour une utilisation passive.

L'utilisation d'un mediaspace n'est pas une activité de premier plan. C'est une activité périphérique qui autorise des interactions spontanées. En contrepartie, c'est cette notion d'activité périphérique qui rend la conception d'interfaces pour le mediaspace si complexe. La fréquence et la variété des usages observés sont en effet difficilement compatibles avec une conception basée sur l'analyse de tâches. L'accès permanent au mediaspace et à ses occupants élimine la notion d'appel ou de connexion avec un début et une fin formellement définis. Cette approche favorise au contraire les transitions entre différents niveaux de conscience de l'activité du groupe, permettant de passer d'*entendre* à *écouter* ou d'*apercevoir* à *regarder*. Lorsque les services et les éléments physiques du mediaspace sont bien intégrés dans l'environnement des utilisateurs, ces transitions sont moins brutales, ce qui réduit le besoin de coordination entre les utilisateurs et le système. L'interaction devient alors plus naturelle, plus concise, plus directe.

En s'inspirant des idées de l'*Informatique disséminée* (*Ubiquitous computing* [Wei91]), B. Buxton et ses collègues de l'Université de Toronto ont mis au point un certain nombre de dispositifs matériels pour faciliter l'intégration du mediaspace dans l'environnement quotidien des utilisateurs [Bux97]. Au lieu de concentrer les interactions avec le système vidéo au niveau d'un unique ensemble moniteur/caméra, ils multiplient les équipements, jouant sur leur taille et leur emplacement dans l'environnement pour leur attribuer des fonctions différentes. La Figure 29 montre ainsi l'utilisation d'une caméra et d'un moniteur dédiés aux connexions de type coup d'œil. La taille et le positionnement de l'ensemble au niveau de la porte du bureau font apparaître les visiteurs du mediaspaces au même endroit et approximativement à la même échelle que les visiteurs du couloir. Ce dispositif n'est pas destiné à remplacer les

équipements traditionnels mais à les compléter pour reproduire une évolution naturelle de la communication : après avoir jeté un coup d'œil depuis la porte, l'utilisateur distant peut établir une connexion classique et sera alors visible sur un moniteur plus proche, placé à côté de l'écran informatique.

Figure 29. Intégration des moyens audiovisuels du mediaspace (Ontario Telepresence Project)

Bien évidemment, la multiplication des équipements audiovisuels peut rendre plus complexe l'utilisation du mediaspace. Comment choisir la caméra ou le moniteur à utiliser ? Une expérience de collaboration à distance assistée d'un système de prises de vue multiples a montré que cette multiplication des points de vue n'est valable que si les participants peuvent se concentrer sur l'activité et non sur le fonctionnement du système permettant de changer de vue [GSHL93]. Dans la mesure du possible, les changements de vues doivent donc être intuitifs et transparents pour l'utilisateur, ne nécessitant aucune action explicite.

Buxton et al. ont également développé un ensemble de technologies qui permettent de contrôler le mediaspace par des moyens physiques naturels plutôt que par une interface logicielle. La Figure 30 montre un exemple de dispositif utilisé pour contrôler l'accessibilité par le mediaspace. Le dispositif consiste en une simple souris à boule fixée à la porte du bureau de l'utilisateur qui permet au système informatique de détecter l'état de cette porte (ouverte, entrouverte ou fermée). L'accès à l'utilisateur par le mediaspace est ainsi régi par les mêmes mécanismes que l'accès physique par la porte de son bureau.

Figure 30. Intégration des moyens de contrôle du mediaspace (Ontario Telepresence Project)

Ces deux exemples de l'Université de Toronto illustrent de façon extrême la différence entre l'idée de mediaspace et les autres systèmes de communication vidéo. Le mediaspace n'est pas une application. On "vit dedans". Il est constamment accessible et intimement lié à l'environnement physique.

Intégration et infrastructure logicielle

Les applications de type visiophone perturbent les activités individuelles de la même façon que les salles dédiées à la visioconférence : les utilisateurs doivent positionner et dimensionner les fenêtres de l'application de communication pour qu'elles soient accessibles sans pour autant masquer les autres applications. Le paradigme téléphonique sur lequel repose ces applications de communication alourdit encore leurs interfaces en imposant des actions explicites pour initier et accepter des connexions.

Plusieurs architectures logicielles ont été proposées pour associer des liaisons audio et vidéo à des environnements collaboratifs. *Mermaid* [WSM+90] et *Jupiter* [CDFN95] permettent par exemple aux participants d'une communication audio/vidéo de partager des documents ou des applications. Cependant, ces architectures imposent une couche intermédiaire entre l'utilisateur et son environnement de travail habituel, ses documents et ses applications : l'accès aux documents et aux applications se fait par l'intermédiaire du système de communication. Cette approche est en contradiction avec l'idée d'interaction avec le mediaspace en arrière-plan et se conjugue mal avec des services de perception périphérique comme les coups d'œil ou les vues d'ensemble.

Comme nous l'avons dit précédemment, le mediaspace n'est pas une application. Son interface logicielle ne doit pas se substituer à l'environnement de travail habituel des utilisateurs mais doit au contraire s'intégrer dans l'environnement logiciel de la même façon que les dispositifs audio/vidéo sont intégrés dans le monde réel. A l'origine de l'Informatique Disséminée, M. Weiser dit "ce n'est que lorsque les choses disparaissent que nous sommes libres de les utiliser sans y penser et que nous pouvons nous concentrer sur de

nouveaux buts" [Wei91]. Comment peut-on faire disparaître l'interface du mediaspace ?

L'approche disséminée/augmentée peut être appliquée pour le logiciel. J. Grudin suggère d'ailleurs que les fonctionnalités collecticielles doivent être intégrées à celles utilisées dans le cadre d'activités individuelles, les ajoutant de préférence à des applications dont le succès est reconnu [Gru94]. Les interactions avec le mediaspace ne doivent pas se faire à travers une unique interface séparée du reste de l'environnement informatique. Le son, l'image et les interfaces doivent être rendus accessibles depuis les applications et documents existants, individuels ou collectifs. Le logiciel d'un mediaspace doit être pensé en termes de composants à intégrer dans les habitudes de travail et non en terme de nouvelle application.

3.2. Flexibilité

> La *flexibilité* mesure la facilité avec laquelle les utilisateurs peuvent adapter un système donné pour qu'il corresponde à leurs besoins particuliers

S'il est vrai qu'un accès permanent, rapide et intuitif aux services du mediaspace est important, il ne suffit pas pour autant à assurer son adoption par les utilisateurs. Ceux-ci agissent généralement en fonction de la situation, ne suivant pas de plan prédéfini mais improvisant et adaptant leurs actions au contexte [Suc83]. L'environnement de communication que constitue le mediaspace se définit donc non seulement par ce qu'il propose mais surtout par la liberté qu'il laisse à l'utilisateur de se l'approprier.

S. Harrison et P. Dourish donnent un exemple qui illustre bien l'importance de ce phénomène d'appropriation dans l'acceptation du mediaspace [HD96]. Ils citent les conclusions de deux rapports, l'un sur le système VideoWindow de BellCore [FKC90] et l'autre sur le premier Media Space de Xerox [OB91]. Alors que ces deux systèmes ont été développés au même moment, avec les mêmes intentions et dans des contextes similaires, les conclusions des deux rapports sont opposées : "il manque quelque chose au système VideoWindow actuel en raison de facteurs que nous n'arrivons pas à identifier" et "le media space a apporté quelque chose de merveilleux à ceux qui ont participé au lien Palo Alto-Portland". Selon Harrison et Dourish, cette différence d'appréciation en faveur du Media Space tient principalement à la nature du matériel utilisé : fixe, encombrant, fragile pour VideoWindow et léger, de faible coût et facilement manipulable pour le Media Space.

La configuration matérielle d'un mediaspace peut effectivement être facilement adaptée en déplaçant les équipements, en les ajustant, en les remplaçant ou en les combinant. Les utilisateurs

peuvent rapidement modifier le volume sonore, régler la netteté de l'image, ajouter une table de mixage ou un objectif grand angle. Cette flexibilité est d'autant plus importante dans le cas d'une activité fortement collaborative. La possibilité de changer rapidement le point de vue distant en déplaçant simplement la caméra locale permet par exemple de se focaliser sur la tâche à accomplir et non sur les aspects techniques de la communication. Cette situation contraste une nouvelle fois avec les salles de visioconférence où les utilisateurs sont généralement limités à un ensemble de configurations prédéfinies qu'ils ne peuvent modifier.

Flexibilité et infrastructure logicielle

Le courrier électronique reste à ce jour l'application collecticielle ayant le plus de succès [Gru94, BD95]. Le protocole sur lequel repose la plupart des échanges de message est pourtant l'un des plus simples qui soient. Mais c'est justement cette simplicité qui est la clé du succès : le protocole ne fait aucune supposition sur le contenu des messages. Le contenu est laissé entièrement libre, ce qui permet d'utiliser le courrier électronique comme pense-bête personnel, comme forum de discussion à plusieurs ou comme interface de commande à des systèmes automatisés [Mac88]. Considérez la façon dont vous gérez votre courrier électronique. L'ordre dans lequel vous composez les différents éléments d'un message (corps du message, destinataire, sujet, etc.), la méthode utilisée pour l'envoyer, par exemple en conservant une copie, la façon dont vous organisez les messages reçus. Tous ces choix vous sont propres. Vous avez construit vous-même votre propre système de gestion du courrier.

Le développement des services des premiers mediaspaces s'est fait selon le même principe de personnalisation. L'utilisation des boutons [MCLM90], par exemple, a permis aux utilisateurs de RAVE de donner des noms différents à des services de base pourtant identiques, comme le visiophone et le partage de bureau, ou à les modifier pour en créer de nouveaux comme le coup d'œil, qui n'est finalement qu'une connexion classique à durée limitée. Cette possibilité de s'approprier les services existants pour les modifier ou les composer pour en créer de nouveaux a contribué à diversifier les usages du mediaspace. Comme le notent [BD95] et [Mac90], permettre la personnalisation, c'est favoriser l'innovation par la co-évolution : la technologie façonne les usages, et donc les utilisateurs et en même temps, ceux-ci se l'approprient de façon non anticipée.

Nous avons choisi de désigner l'ensemble des logiciels nécessaires au fonctionnement d'un mediaspace par l'expression *infrastructure logicielle*. Ce choix n'est pas innocent. Nous aurions pu utiliser l'expression architecture logicielle, mais ces deux mots n'ont pas la même signification [DUF] :

```
infrastructure (n.f.) :

  Ensemble des installations nécessaires à une activité,
  à la vie en un lieu.

architecture (n.f.) :

  Art de construire des édifices selon des proportions et
  des règles déterminées par leur caractère et leur
  destination.

  Structure, principe d'organisation.
```

Le mediaspace ne doit pas être un système clos régit par des principes rigides, mais un environnement ouvert et personnalisable. Nous devons donc bien parler d'infrastructure logicielle. Les "installations" nécessaires à la vie dans le mediaspace sont les services de base nécessaires à l'établissement des connexions audio/vidéo. Mais le plus important n'est pas cet ensemble de départ, mais les possibilités offertes pour le développer.

Dans de nombreux systèmes, la flexibilité n'est accessible qu'aux programmeurs ou aux utilisateurs experts. Les boutons de RAVE étaient suffisamment simples pour pouvoir être utilisés par toutes les classes d'utilisateurs. La conception de l'infrastructure logicielle du mediaspace doit prendre en compte ces différences entre les utilisateurs et essayer, dans la mesure du possible, de rendre la flexibilité accessible à la majorité des utilisateurs, qui ne savent pas ou ne souhaitent pas programmer.

3.3. Intimité

> L'*intimité*[11] mesure le degré de facilité avec lequel les utilisateurs peuvent comprendre, utiliser et faire confiance aux mécanismes qui contrôlent l'information publiée les concernant ainsi que leur disponibilité pour les autres utilisateurs

Fish et al décrivent trois volontés qui entrent en compétition lors de l'utilisation d'un système de communication vidéo : la volonté d'être accessible, la volonté de contrôler sa disponibilité vis-à-vis des autres utilisateurs et la volonté de contrôler l'information disponible à propos de soi [FKRR92]. La difficulté à trouver un compromis entre ces trois volontés est encore augmentée par l'utilisation inévitable de moyens technologiques pour contrôler

[11] Nous avons choisi ce mot pour traduire le mot anglais *privacy*. [DUF] définit le mot *intimité* par «Caractère de ce qui convient au confort de la vie intime» et le mot *intime* par «Qui est tout à fait privé» ou «Qui ne réunit que des proches».

l'information disponible et l'accès à cette information par d'autres utilisateurs.

Le phénomène de confiance peut se définir comme l'augmentation de sa vulnérabilité à travers une entité tierce que l'on ne contrôle pas [Roc98]. L'utilisation des moyens de contrôle du mediaspace constitue un acte de confiance de la part de l'utilisateur envers le système. La gestion des problèmes de protection de la vie privée au sein du mediaspace passe donc nécessairement par la confiance des utilisateurs dans les moyens techniques mis en œuvre.

Lorsqu'un utilisateur débranche ou éteint un équipement audiovisuel du mediaspace, il connaît à l'avance les effets de ce geste sur l'équipement en question ainsi que sur l'ensemble du système. Il sait ainsi que s'il débranche une caméra, elle ne pourra plus filmer, ce qui empêchera à coup sur les autres personnes de le voir. Les effets de ces actions directes sur le matériel peuvent être vérifiés de façon assez simple, en connectant la caméra débranchée à un moniteur par exemple, et il est généralement possible de revenir à l'état antérieur du système en inversant le geste effectué (ici, en rebranchant la caméra).

Intimité et infrastructure logicielle

La possibilité d'action directe sur les équipements du mediaspace constitue pour les utilisateurs le moyen le plus simple et le plus sur pour gérer leur vie privée. Mais comme nous l'avons fait remarquer dans notre évocation du premier Media Space, il est intéressant de pouvoir faire la distinction entre un matériel en panne et un utilisateur qui veut être laissé tranquille. Pour être utilisés, les moyens de contrôle logiciels du mediaspace doivent donc présenter les mêmes propriétés que l'action directe sur le matériel - simplicité, contrôlabilité et réversibilité - afin que les utilisateurs leur accorde la même confiance[12].

Le contrôle d'accès logiciel se fait généralement par l'attribution de droits de lecture et d'écriture. Cette conception est liée à la nature généralement statique des données sur lesquelles porte le contrôle. La protection de la vie privée dans un mediaspace est un problème complexe en raison de la nature dynamique de l'audio et de la vidéo. Dix secondes de vidéo n'ont pas la même valeur en fonction du nombre de personnes présentes à l'image et de leur activité. Ces deux paramètres, choisi parmi de nombreux autres, varient tout au long de la journée. Il est donc impossible de déterminer une politique de contrôle d'accès uniquement en fonction des médias utilisés.

[12] Faisant ainsi référence à l'importance de la confiance entre les utilisateurs eux-mêmes et le système, W. Buxton propose l'expression *trustification technologies* plutôt que *communication technologies* pour décrire ses travaux sur la téléprésence (http://www.ntticc.or.jp/pub/ic_mag/ic025/pdf/054-061e.pdf)

Le mediaspace doit fournir aux utilisateurs le maximum d'indications sur l'identité de la personne qui demande un service et ses intentions. La diversité des services aide à la classification des demandes en termes d'intention et participe donc aux mécanismes de contrôle d'accès. Les choix offerts par le système ne doivent pas se limiter à des propositions binaires, tels qu'accepter ou refuser la demande, mais doivent permettre de décrire des réponses plus complexes offrant une possibilité de négociation. Tous ces éléments contribuent à créer un sentiment d'accessibilité sélective [Dou93]. Diverses notifications visuelles ou auditives doivent pouvoir être associées à la réception, l'exécution et la fin d'une requête afin que l'état local du système soit toujours connu. Enfin, ces mécanismes de contrôle et de notification doivent pouvoir être automatisés afin de limiter les demandes de confirmation et de permettre aux utilisateurs la création de différentes politiques de contrôle d'accès interchangeables. La flexibilité est donc une fois encore indispensable, véritable clef du compromis entre des interactions spontanées et naturelles et un contrôle d'accès explicite.

4. Résumé du chapitre

L'évolution des systèmes de communication vidéo commerciaux les a fait se rapprocher progressivement vers l'utilisateur. Des salles dédiées à la visioconférence, nous sommes passés aux systèmes mobiles de téléréunion pour des groupes de plus petite taille, puis à l'ordinateur personnel communicant. Cette évolution est actuellement au stade ou les travaux sur les mediaspaces ont commencé : du matériel audiovisuel associé à un équipement informatique devant chaque utilisateur.

Les premiers mediaspaces ont montré qu'il est possible d'utiliser les moyens de communication vidéo pour des échanges plus variés que ceux proposées par les systèmes de visioconférence, téléréunion ou visiophone. Cette diversité des usages couvre à la fois des situations traditionnelles de collaboration formelle mais également des formes plus subtiles de communication facilitant la coordination et permettant des rencontres ou discussions informelles entre les utilisateurs.

La généralisation de ce type d'environnement s'est heurtée à des problèmes techniques liés à la lourdeur et à la spécificité des infrastructures analogiques utilisées. En outre, le succès du déploiement d'un mediaspace est largement conditionné par le contexte social dans lequel il est introduit. La motivation initiale des utilisateurs et leur implication dans la définition et l'évolution des services sont donc souvent plus importants que les aspects techniques.

L'avenir de l'ordinateur personnel communicant est selon nous lié à celui des mediaspaces. L'un a besoin de nouveaux usages et de nouvelles formes d'interaction, les autres ont besoin d'un changement de technologie pour pouvoir se développer. Tous deux ont besoin d'un contexte social favorable. La situation actuelle semble être propice à l'apparition d'une deuxième génération de mediaspaces, qui bénéficieraient à la fois des progrès techniques pour s'affranchir de l'infrastructure analogique et d'un changement d'attitude vis à vis des moyens de communication personnels pour faciliter leur acceptation.

Nous avons identifié trois dimensions essentielles des premiers mediaspaces qui ont facilité leur mise en œuvre et leur acceptation :

- l'intégration du système de communication dans les habitudes des utilisateurs, qui leur offre un moyen d'accès permanent, rapide et intuitif aux autres personnes ;

- sa flexibilité, qui leur permet d'adapter le système à leurs besoins particuliers ;

- son intimité, ou la facilité avec laquelle ils comprennent, utilisent et font confiance aux mécanismes de contrôle d'accès.

L'intégration du mediaspace dans les habitudes de travail des utilisateurs n'a pu se faire que grâce à sa flexibilité et à son intimité. Nous avons montré comment chacune de ses trois caractéristiques peut se traduire en termes d'infrastructure logicielle. Dans le chapitre suivant, nous poursuivons notre étude sur l'intégration de la vidéo dans les systèmes informatiques interactifs en abordant la délicate question du choix de l'infrastructure logicielle pour la réalisation de nouveaux mediaspaces.

Chapitre III

Le Web comme infrastructure logicielle pour les Mediaspaces

Dans le chapitre précédent, nous avons présenté trois caractéristiques essentielles des premiers mediaspaces qui ont facilité leur mise en œuvre et leur acceptation : leur intégration dans les habitudes des utilisateurs, leur flexibilité et la garantie d'intimité qu'ils assurent. Ces caractéristiques étaient en partie liées à l'utilisation de matériel analogique. L'accès rapide et permanent au système était ainsi favorisé par l'utilisation d'un moniteur séparé et dédié au mediaspace. Dès lors que l'on envisage la conception d'un système entièrement numérique, le choix de l'infrastructure logicielle pour réaliser l'interface et le cœur du système devient fondamental puisqu'il va lui falloir éventuellement compenser la perte de ces propriétés liées au matériel en proposant par exemple d'autres moyens d'accéder rapidement au système, quelle que soit l'activité en cours.

Dans ce chapitre, nous proposons l'utilisation du Web comme infrastructure logicielle pour les mediaspaces. Nous expliquons pourquoi nous avons choisi cette infrastructure plutôt que d'autres, nous présentons les technologies qui sont à la base de son développement, puis nous montrons en quoi ces technologies sont adaptées à la réalisation d'un environnement de type mediaspace et de ses interfaces.

1. Choix de l'infrastructure : une affaire de public

Comme nous l'avons vu au chapitre précédent, l'infrastructure logicielle d'un mediaspace doit répondre aux trois importantes nécessités d'intégration, de flexibilité et de garantie d'intimité. Si

cette dernière notion est principalement liée au fonctionnement interne du système et à la confiance que les utilisateurs lui accordent, la flexibilité et l'intégration dépendent largement de l'infrastructure et des outils utilisés pour la réalisation des interfaces et des services du mediaspace.

La nécessité de flexibilité du logiciel n'est pas propre au mediaspace, mais se retrouve dans la plupart des applications collaboratives [BD95]. Plusieurs méthodes de conception ont été proposées pour permettre l'adaptation de ces applications par leurs utilisateurs, dont l'utilisation de *protocoles ouverts* [RG93] ou de *composants logiciels* [Sti97, tH98]. Cette notion de composant logiciel est inspirée de l'utilisation d'éléments standards en électronique ou en mécanique comme les transistors, résistances, vis ou boulons qui permettent, par composition, de créer des objets complexes. Un composant logiciel est défini comme "une unité de composition à interfaces contractuelles et dépendance contextuelles explicites" [SP96].

Cette approche composant est particulièrement séduisante dans le cadre du mediaspace. Il serait en effet intéressant de pouvoir développer un ensemble d'éléments logiciels de base pour les programmeurs et les utilisateurs, chaque élément correspondant à un besoin ou à une fonctionnalité du système : gestion de session, gestion des connexions analogiques, gestion des flux numériques, contrôle d'accès, notification, etc. Les utilisateurs pourraient alors définir leurs propres services en configurant, remplaçant ou combinant ces éléments comme ils le font avec les dispositifs physiques.

Si une approche basée sur des composants modulaires semble bien adaptée à la conception logicielle d'un mediaspace, la nécessité d'intégration à l'environnement informatique existant exclue la création d'une nouvelle infrastructure de composants et oblige à choisir parmi celles existantes. Il existe en effet plusieurs infrastructures permettant de développer des composants logiciels réutilisables, parmi lesquelles on peut citer CORBA[13], COM[14] ou JavaBeans[15]. Celles-ci présentent cependant un inconvénient majeur : elles ont été conçues pour un public de programmeurs, et non d'utilisateurs[16]. En conséquence, la conception, l'implémentation, la configuration et la composition de leurs éléments sont difficilement accessibles aux utilisateurs ne sachant

[13] http://www.corba.org/

[14] http://www.microsoft.com/com/

[15] http://java.sun.com/beans/

[16] La spécification de CORBA est, à ce sujet, assez représentative : le mot *user* (utilisateur) y est partout employé pour désigner le concepteur du composant, et non celui qui s'en sert.

pas programmer, si l'on excepte la simple inclusion d'un élément dans un autre.

L'un des précurseurs, aujourd'hui abandonné, de ces systèmes à base de composants est OpenDoc[17]. A la différence des autres systèmes mentionnés, OpenDoc bénéficiait d'une réelle conception centrée sur l'utilisation des composants et non seulement sur leur implémentation. Ce système est à l'origine de la vision centrée sur des documents composites construits et manipulés par l'utilisateur, opposée à la vision classique centrée sur les applications utilisées à ces fins.

Un mediaspace est un système qui permet à ses utilisateurs de communiquer. L'importance de l'intégration et de la flexibilité du mediaspace concerne donc avant tout ses utilisateurs. L'approche basée sur l'utilisation de composants parait séduisante, mais les systèmes de composants existants sont trop éloignés des utilisateurs pour pouvoir être envisagés. L'infrastructure idéale pour le mediaspace devrait permettre à la fois sa conception en termes de composants, son intégration aux documents et applications existants, et son adaptation à la fois par ses concepteurs et ses utilisateurs. Nous allons voir que le Web répond à toutes ces exigences, mais avant cela, commençons par préciser ce qu'il est réellement.

2. Le Web

Au début des années 90, T. Berners-Lee et ses collègues du CERN ont fait un constat assez simple : si l'on veut rendre accessible au plus grand nombre une information, il faut être capable de la désigner, de décrire son organisation et de la transmettre. Partant de ce constat, ils ont conçu un schéma d'adressage, un langage de description de documents hypermédia et un protocole d'échange qui constituent aujourd'hui encore la base du *World Wide Web* [BLCL+94], que nous désignons maintenant par les termes WWW, W3 ou tout simplement : le Web.

Grâce à ces trois standards et à la multiplication des données et services disponibles, l'ordinateur personnel s'est transformé en point d'accès à l'information. Cette étape marque l'avènement du *navigateur*, l'application graphique interactive qui permet d'accéder rapidement et simplement aux informations locales ou distantes. En quelques années seulement, le navigateur Web est devenu un élément incontournable de l'environnement informatique quotidien, se substituant à de nombreux outils de gestion de fichiers ou de messagerie électronique. Aujourd'hui, le Web est un moyen de communication de masse qui vient compléter la presse, la radio, la

[17] http://developer.apple.com/techpubs/mac/ODProgGuide/

télévision ou les panneaux d'affichages. Chaque jour ou presque, il se transforme et devient plus grand, plus beau, plus rapide, plus dynamique, plus interactif et plus personnalisable.

Le Web est un ensemble de ressources accessibles par Internet. Ces ressources peuvent être des objets, tels qu'un document électronique ou une image, des services, comme "le temps qu'il fait à Bonn", ou bien encore des collections d'autres ressources. Si l'on veut comprendre comment fonctionne le Web et comment l'utiliser pour construire de nouveaux services, il faut comprendre comment ces ressources sont nommées, décrites et transférées.

2.1. Désigner les ressources par des URIs

Berners-Lee et al. ont défini le concept d'*URI* (*Uniform Resource Identifier* [BLFM98]) qui regroupe toutes les formes syntaxiques permettant de désigner une ressource de manière non ambiguë. Une URI est une chaîne de caractères qui permet d'identifier une ressource abstraite ou physique. Il existe deux catégories d'URIs : celles qui nomment les ressources et celles qui font référence à un mécanisme d'accès particulier. Les premières sont qualifiées d'*Uniform Resource Name*, ou *URN*, et les secondes d'*Uniform Resource Locator*, ou *URL*.

Le concept d'URI ne se limite pas uniquement aux ressources accessibles par le réseau informatique. Un livre peut par exemple être désigné par son numéro ISBN ou une phrase du type "le livre en haut à gauche". En termes d'URI, la première désignation est une URN tandis que la seconde est une URL. Un schéma d'URN a d'ailleurs été défini pour la numérotation ISBN[18]. L'intérêt des URLs est de permettre éventuellement l'accès immédiat à la ressource - si l'on sait où est la bibliothèque - tandis que les URNs assurent une désignation perpétuelle, même si la ressource est déplacée, devient temporairement inaccessible ou disparaît.

La syntaxe proposée par Berners-Lee et al. pour les URIs est uniforme. Cela signifie que des identificateurs différents peuvent être utilisés dans le même contexte même si les mécanismes qu'ils impliquent sont différents. On peut ainsi dire "je voudrais le grand livre là-bas" ou "je voudrais *Contes et légendes de la Grèce antique*". Cela permet également une interprétation sémantique des conventions syntaxiques communes à différents types d'identificateurs. Cela permet ainsi d'isoler du reste de l'URI le nom de la machine qui possède ou sert la ressource, quel que soit le protocole utilisé pour l'obtenir. Enfin, cela permet également d'introduire de nouveaux types d'identificateurs sans perturber l'utilisation de ceux existants.

[18] Je vous suggère d'ailleurs la lecture de `URN:ISBN:0-849-30086-X`

2.2. Transmettre des ressources par HTTP

Le protocole *HTTP* (*Hypertext Transfer Protocol* [FGM+99]) permet à des applications d'échanger les données correspondant aux ressources du Web. Deux versions de ce protocole coexistent aujourd'hui : HTTP/1.0, la version la plus utilisée, et HTTP/1.1 qui possède quelques possibilités supplémentaires.

HTTP est un protocole client-serveur sans état basé sur TCP/IP : le client envoie une requête au serveur, généralement à la suite d'une action d'un utilisateur, puis le serveur répond à la requête indépendamment des requêtes précédentes. La spécification du protocole propose un ensemble de méthodes et de codes de retour qui permettent aux clients et aux serveurs d'exprimer leurs requêtes et réponses de façon standard. Les méthodes proposées permettent par exemple à un client de demander ou d'envoyer des données à un serveur. Les méthodes `GET` et `POST` sont ainsi utilisées pour consulter des documents et saisir des formulaires. Les codes de retour permettent à un serveur de répondre positivement à une requête, par exemple `200 OK` suivi des données, mais également de demander une autorisation d'accès ou un paiement, de rediriger la requête vers une autre ressource, ou encore de signaler une erreur, comme le fameux `404 Not Found`.

Une URL HTTP permet à un client d'accéder à une ressource par le protocole HTTP. Pour cela, le client a besoin du nom ou de l'adresse IP de la machine sur laquelle tourne le serveur, du numéro de port TCP utilisé, ainsi que du nom de la ressource sur le serveur, qui est généralement un chemin d'accès. Lorsque l'URL comporte toutes ces informations, elle est dite absolue. Dans le cas contraire, l'URL est dite relative. L'URL peut également spécifier un certain nombre d'options concernant le traitement de la requête par le serveur puis le traitement de sa réponse par le client (Figure 31).

```
http://machine:port/ressource[?options_serveur][#option_client]
```

Figure 31. Schéma d'URL HTTP absolue

Lorsqu'une application doit accéder à une ressource spécifiée par une URL HTTP, elle décode celle-ci selon le format de la Figure 31, ouvre une nouvelle connexion TCP ou utilise une connexion existante vers le serveur, envoie la requête appropriée concernant la ressource avec les options pour le serveur et attend une réponse. Il est important de noter que la méthode à utiliser n'est pas codée dans l'URL.

La Figure 32 montre le format d'une requête HTTP. Les couples `clé : valeur` optionnels peuvent être utilisés à titre d'information ou pour modifier la sémantique de la requête, pour demander par exemple que le document référencé ne soit transmis que s'il a changé depuis une certaine date. Contrairement aux options

décrites dans l'URL, celles-ci sont le plus souvent déterminées par l'application cliente et non par son utilisateur et concernent l'utilisation du protocole HTTP et non la ressource considérée.

```
méthode /ressource[?options_serveur] version_HTTP
[clé: valeur]*
```

Figure 32. Format d'une requête HTTP

La Figure 33 montre le format d'une réponse. Comme pour la requête, les champs optionnels peuvent compléter l'information fournie par le code de retour ou modifier la sémantique de la réponse.

```
version_HTTP code explication
[clé: valeur]*

[données]
```

Figure 33. Format d'une réponse HTTP

2.3. Relier des ressources avec HTML

Le langage *HTML* (*HyperText Markup Language* [RHJ99]) permet de créer des documents hypermédia reliant entre elles différentes ressources. Un document HTML contient du texte simple et des marques, appelées aussi balises, qui définissent la structure logique du texte (Figure 34).

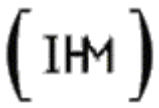
```
<html>

<head>
<title>Exemple simple</title>
</head>

<body>
<img src="http://www-ihm.lri.fr/~roussel/img/ihm.gif">
<a href="http://www-ihm.lri.fr/">
  Groupe Interaction Homme-Machine du LRI
</a>
</body>

</html>
```

Figure 34. Exemple de code HTML

Ainsi, dans l'exemple ci-dessus, `<html> … </html>` marque le début et la fin du document, `<head> … </head>` délimite la section des meta-informations - ici, le titre - et `<body> … </body>` délimite la section des informations affichables. Un certain nombre de balises HTML permettent de lier le document à d'autres ressources décrites par des URLs. La balise `<img>` permet ainsi

d'insérer une image dans le document, tandis que la balise `<a> … </a>` crée un lien hypertexte vers la ressource spécifiée, en l'occurrence, le site Web du groupe IHM du LRI.

De tels documents HTML peuvent être stockés sur un système de fichiers ou générés à la demande par un serveur HTTP. Les applications clientes sont chargées de la présentation des informations à l'utilisateur (Figure 35).



Groupe Interaction Homme-Machine du LRI

Figure 35. Exemple de présentation du document décrit par le code HTML de la Figure 34

Le langage HTML est souvent perçu par les débutants du Web comme un langage de programmation. Le langage lui-même ne comporte pas de structure de contrôle et n'a pas grand chose à voir avec un véritable langage de programmation. En revanche, HTML permet d'inclure des scripts dans les documents pour les rendre interactifs. Ces scripts peuvent modifier le contenu du document dynamiquement ou contrôler la saisie de formulaires. Ils peuvent être déclenchés lors de l'affichage du document ou lorsque l'utilisateur effectue certaines actions comme l'utilisation du clavier ou de la souris.

3. Le Web comme infrastructure logicielle pour les mediaspaces

La plupart des applications que nous utilisons aujourd'hui sont capables de lire ou de produire des fichiers HTML, d'interpréter des URIs et d'échanger des données par HTTP. En quelques années, ces trois standards ont considérablement facilité l'échange de documents à distance. Pourtant, bien que nous l'utilisions de plus en plus fréquemment pour échanger du texte, des images ou des sons avec d'autres individus, le Web reste principalement un point d'accès à l'information, un *lien entre l'individu et l'information*.

Le Web propose des mécanismes pour décrire, transmettre et relier des ressources. Toutefois, de manière comparable à X Window dans le domaine des interfaces graphiques, il n'impose aucune politique d'utilisation particulière de ces mécanismes. Dans cette section, nous explorons les possibilités offertes par ceux-ci pour créer une infrastructure logicielle et des interfaces adaptées à la *communication entre individus*. Nous allons montrer comment il est possible d'utiliser le protocole HTTP pour contrôler un réseau audio/vidéo analogique ou transmettre de la vidéo numérique. Toutefois, nous n'évoquerons pas les problèmes de qualité de transmission, de latence ou de perte de paquets qui sont

habituellement abordés dès que l'on s'intéresse à l'utilisation d'audio ou de vidéo sur Internet. Sans pour autant nier l'importance de ces problèmes liés aux protocoles et réseaux utilisés, nous nous intéressons plus spécialement aux usages combinés de la vidéo et du Web et plus précisément à l'utilisation de ce dernier comme infrastructure logicielle pour les mediaspaces. Nous allons donc montrer comment les standards à l'origine du Web permettent de concilier intégration, flexibilité et intimité.

3.1. Intégration

La plupart des applications utilisent HTTP pour échanger des données. Mais il est également possible d'utiliser ce protocole pour d'autres formes de communication. L'un des codes de retour permet par exemple au serveur d'envoyer un simple acquittement au client pour lui indiquer que sa requête a été traitée, mais qu'elle n'a généré aucune nouvelle information : `204 No content`. Ce type de dialogue entre le client et le serveur permet d'utiliser HTTP comme une interface de commande et de contrôle.

Cette possibilité permet à son tour d'envisager le contrôle à distance d'équipements audiovisuels. Une matrice de commutation analogique peut ainsi être gérée par un serveur HTTP au moyen de ressources du type `/connecte_X_avec_Y` ou `/connecte/X/Y`.

HTTP est à l'origine un protocole sans état : les serveurs répondent à chaque requête indépendamment des précédentes. Un mécanisme a été ajouté à HTTP pour permettre aux clients et aux serveurs de situer leurs requêtes et réponses dans un contexte commun. Ce mécanisme, connu sous le nom de *cookies* [KM97], utilise les champs optionnels prévus dans les requêtes et les réponses pour transmettre l'information nécessaire pour retrouver le contexte commun. Lorsqu'un client reçoit une réponse accompagnée d'un cookie, il est supposé transmettre le cookie avec toutes les requêtes envoyées par la suite à ce serveur. L'information contenue dans le cookie est généralement un identificateur numérique unique qui permet au serveur d'identifier le client.

L'utilisation de cookies permet de simplifier la gestion des connexions analogiques évoquée précédemment. En identifiant les utilisateurs au moyen de cookies, il est possible d'utiliser des ressources du type `/appelle/Y` au lieu de `/connecte/X/Y`. La même ressource, et donc la même URL, peut alors servir à différents utilisateurs pour se connecter à un des nœuds du mediaspace.

HTTP utilise le format *MIME* (*Multipurpose Internet Mail Extensions* [FB96]) pour décrire le type des données éventuelles contenues dans la réponse à une requête. Généralement, une réponse HTTP ne comporte qu'un seul bloc de données. MIME permet cependant de coder plusieurs blocs de données au sein d'une

unique réponse. Cette possibilité repose sur l'utilisation d'un type MIME standard, le type `multipart/mixed`. Les différents blocs de données sont alors transmis à la suite, un délimiteur permettant au client de les séparer à la réception. Le *server push* [Net95] est une variante de ce mode de transfert introduite par Netscape en 1995. Le type MIME correspondant au server-push est `multipart/x-mixed-replace`[19]. Le mot `replace` indique que chaque nouveau bloc de données ne vient pas compléter le précédant mais doit le remplacer. Le client étant censé présenter le dernier bloc reçu, le serveur peut envoyer chaque nouveau bloc de données quand il le veut, à la vitesse où il le veut. C'est le seul cas de figure ou l'initiative du transfert de données est du côté du serveur, ce qui explique le nom de server-push.

Dans le cadre du mediaspace, le mécanisme de server-push peut être utilisé pour transmettre une séquence d'images, chaque image venant remplacer la précédente (Figure 36). Cette transmission se faisant par une simple extension du protocole HTTP, toute application implémentant l'extension et sachant afficher une image pourra afficher des séquences vidéo capturées en temps réel ou préenregistrées et transmises par le serveur. Dans le contexte du mediaspace, cela signifie que l'accès aux images des caméras peut se faire sans logiciel spécifique.

[19] Le `x-` de `multipart/x-mixed-replace` indique que ce type est expérimental et n'a pas été standardisé. C'est sans doute ce qui explique que Microsoft ne l'a jamais implémenté dans ses différentes versions d'Explorer...

```
HTTP/1.0 200 OK
Content-type: multipart/x-mixed-replace;boundary=-->

-->
Content-Type: image/jpeg
Content-Length: longueur du bloc de données

données de la première image
-->
Content-Type: image/jpeg
Content-Length: longueur du bloc de données

données de la deuxième image
-->
Content-Type: image/jpeg
Content-Length: longueur du bloc de données

données de la troisième image
-->
...
```

Figure 36. Principe de la transmission d'une séquence d'images JPEG utilisant le mécanisme de server-push

L'intégration du mediaspace dans les habitudes des utilisateurs nécessite un accès simple et rapide à ses services. En combinant le mode commande, les cookies et le server-push, il est possible de réaliser des serveurs HTTP destinés à la gestion des connexions d'un mediaspace analogique et à la transmission d'images numériques. Ces services pouvant être décrits par des URLs, ils peuvent être rendus accessibles depuis n'importe quel document HTML en y ajoutant des liens vers les serveurs du mediaspace. De plus en plus de librairies de communication permettent d'envoyer des requêtes HTTP depuis les principaux langages de programmation utilisés (C/C++, Java, Tcl/Tk, Python, Perl, etc.). En plus de l'accès par les documents HTML, les services d'un mediaspace basé sur HTTP sont donc accessibles à toute application écrite dans l'un de ces langages.

3.2. Flexibilité

Dès lors que les services du mediaspace sont accessibles par HTTP, les clients peuvent utiliser les paramètres optionnels de la requête pour décrire plus précisément le service à exécuter. Ces options peuvent contrôler la durée d'une connexion analogique (par exemple, `http://mediascape/glance.nicolas?duration=3`) ou la résolution des images numériques transmises (par exemple, `http://mediascape/glance.nicolas?zoom=4`).

En plus des rôles de client et serveur, HTTP introduit la notion d'intermédiaire. Un intermédiaire se comporte à la fois comme un client et un serveur : il reçoit des messages, requêtes ou réponses, réécrit tout ou partie du message et le fait suivre vers le destinataire initial. Clients, intermédiaire et serveurs peuvent donc être

composés pour former une chaîne de requêtes/réponses. En ajoutant des intermédiaires, les concepteurs d'un système basé sur HTTP peuvent faire évoluer celui-ci progressivement. Ils peuvent reconfigurer ou remplacer chaque élément de la chaîne indépendamment des autres. La possibilité pour un serveur de rediriger un client vers une autre ressource (par exemple, `303 See Other`) augmente encore cette modularité. Enfin, les programmeurs peuvent partager leur expérience en s'échangeant des éléments de la chaîne.

Dans le cadre du mediaspace, différents intermédiaires peuvent par exemple implémenter des politiques de contrôle d'accès différentes. Ils peuvent également être utilisés pour passer à travers des protections réseau, un *firewall* par exemple, ou pour connecter deux systèmes de communication vidéo utilisant des formats différents.

Comme nous l'avons vu au chapitre précédent, l'interface de RAVE utilisait des boutons que les utilisateurs pouvaient personnaliser, placer sur leur bureau et s'échanger par courrier électronique. Certaines caractéristiques de ce système décrites dans [MCLM90] se retrouvent dans la façon dont les utilisateurs créent et modifie des documents HTML aujourd'hui ; plusieurs classes d'individus - utilisateur, bricoleur, programmeur - partagent leurs compétences en demandant, empruntant ou volant des bouts de code HTML aux autres utilisateurs. L'architecture du Web encourage ces formes de réutilisation, permettant à tout utilisateur de sauvegarder localement le code HTML de ce qu'il voit dans son navigateur, puis de l'éditer ou d'en extraire certaines parties sans nécessairement comprendre les détails du code. L'extrême tolérance des navigateurs actuels concernant l'utilisation des balises facilite également cette appropriation : certaines balises de fin peuvent être omises, et certains navigateurs acceptent les balises croisées. L'utilisation de documents HTML pour la réalisation d'interfaces au mediaspace permet de profiter de cette culture existante de l'appropriation. Les interfaces du mediaspace peuvent alors être plus simples à apprendre, à utiliser et à personnaliser.

3.3. Intimité

L'un des premiers dessins humoristiques à propos d'Internet montrait un chien en train d'utiliser un ordinateur et disant à un de ses congénères : «sur Internet, personne ne sait que tu es un chien» (Figure 37). Ce dessin illustre bien l'impression d'anonymat que l'on a lorsqu'on navigue sur le Web. Pourtant, il est assez facile pour un serveur HTTP d'obtenir des informations concernant les

personnes qui lui envoient des requêtes, sans que celles-ci en soient nécessairement informées[20].

Figure 37. " *On the Internet, nobody knows you're a dog* "
(dessin de P. Steiner paru dans *The New Yorker* le 5 juillet 1993)

Dès que la connexion TCP est établie entre le client et le serveur HTTP, ce dernier connaît l'adresse IP et éventuellement le nom de la machine d'où provient la requête. En interrogeant un serveur *WHOIS* [HSF85], le serveur HTTP peut obtenir des informations concernant l'organisation qui gère la machine distante. Il peut également essayer de contacter un serveur d'identification sur la machine distante, comme l'*Ident daemon* [Joh85], pour obtenir le nom d'utilisateur de la personne distante. Ce nom d'utilisateur peut ensuite être utilisé pour obtenir plus de détails sur l'utilisateur, comme son nom et son adresse électronique grâce aux protocoles *SMTP* [Pos82] ou *Finger* [Zim91]. La Figure 38 donne un exemple d'informations ainsi obtenues à partir d'une simple connexion TCP.

```
Remote host     : sgi5.lri.fr
Remote TCP port : 7832

Whois lri.fr
  Laboratoire de Recherche en Informatique, LRI
  UA 410 du CNRS Bat 490 UPS, F-91405 Orsay CEDEX

Login name      : roussel
Real name       : Nicolas Roussel
Email address   : roussel@lri.fr

Finger roussel@lri.fr
  Login name: roussel         In real life: Nicolas Roussel
  Directory: /u/roussel       Shell: /bin/sh
  On since Sep  7 10:12:11 on ttyq0 from :0.0
  4 minutes 3 seconds Idle Time
  No unread mail
  Plan:
    See my web page (http://www-ihm.lri.fr/~roussel/)
```

[20] Le site Web de la CNIL (http://www.cnil.fr/) et celui d'Anonymizer (http://www.anonymizer.com/) proposent des démonstrations de ce que nous décrivons ici

Figure 38. Exemple d'informations concernant un utilisateur distant obtenues à partir d'une connexion TCP

En plus des informations sur l'utilisateur fournies par la connexion TCP, la requête HTTP contient de nombreux détails concernant le client utilisé. Parmi ceux-ci, on trouve généralement le nom du système d'exploitation de la machine, le nom et la version du logiciel, les types de données et de codage qu'il sait traiter, l'URL de la dernière ressource consultée, et éventuellement un cookie ou une autorisation d'accès liés à de précédentes requêtes (Figure 39).

```
User-Agent : Mozilla/4.61C-SGI [en] (X11; I; IRIX64 6.5 IP30)
Accept : image/gif, image/jpeg, image/png, */*
Accept-Encoding : gzip
Accept-Charset : iso-8859-1,*,utf-8
Accept-Language : en
Referer : http://www-ihm.lri.fr/
Cookie : USERLOGIN="roussel"
Authorization: Basic bWJsOndlbmR4
```

Figure 39. Exemple d'informations additionnelles transmises par un client dans une requête HTTP

A la lumière de ces exemples, nous réalisons que non seulement il est possible de savoir que l'on est un chien, mais qu'il est même probable que les sites que nous consultons régulièrement connaissent déjà notre nom et notre marque préférée de biscuits[21]... Cette révélation est inquiétante et fait d'ailleurs l'objet d'une série de travaux du Web Consortium visant à préserver la vie privée des internautes[22]. Cependant, elle repose sur un préjugé important concernant la place de l'utilisateur dans les échanges HTTP.

Dans la majorité des applications du Web en effet, l'utilisateur est supposé être du côté du client. Sans doute à l'origine de cette situation, la spécification du protocole HTTP décrit un "agent utilisateur" comme "le client qui est à l'origine d'une requête". La conséquence de cette vision de la place de l'utilisateur est que le serveur n'est en général qu'un outil utilisé de façon collective pour rendre des informations accessibles sur le réseau. Alors que nous parlons souvent de "mes pages Web" ou de "mon site Web", nous parlons généralement du "serveur Web de mon laboratoire" ou du "serveur Web de mon équipe" et très rarement de "mon serveur Web". Le serveur étant perçu comme un outil ou une machine, et le client étant implicitement un utilisateur, les craintes pour la vie privée vont naturellement vers le client.

---

[21] Voir http://www.anonymiser.com/images/newdog.gif pour un autre dessin humoristique allant dans ce sens

[22] Le Web Consortium est l'organisme en charge de l'évolution des protocoles et langages qui constituent le Web. Pour plus de détails sur les travaux mentionnés, voir http://www.w3.org/PICS/

Dans certaines occasions pourtant, le client n'est pas un utilisateur. C'est le cas notamment avec les nombreux robots qui indexent le contenu du Web. Dans ces situations, le souci de protection de vie privée se porte alors du côté du serveur, où sont stockées les données des utilisateurs. Qui n'a pas eu un jour la surprise de voir indexé un document qu'il pensait inaccessible ? Qui ne s'est pas retrouvé inscrit dans une liste de diffusion sans l'avoir demandé, uniquement parce que son adresse électronique figurait sur un document accessible à tout le monde ? Aujourd'hui, mettre un document sur le Web revient souvent à le poser au milieu d'une route en espérant que quelqu'un va le voir. Il est difficile de savoir si le document est effectivement consulté, et si c'est le cas, il est rarement possible de savoir par qui.

Le serveur Web étant un outil collectif, les informations concernant sa configuration et son activité sont difficilement accessibles aux utilisateurs. La configuration utilise souvent une syntaxe peut explicite et même lorsque l'activité du serveur est enregistrée dans des fichiers (logs), il est difficile pour les utilisateurs d'avoir une idée de la part de cette activité les concernant. Pour obtenir cette information, ils doivent en effet savoir où sont stockés les fichiers, comment ils sont formatés, et ils doivent surtout avoir les connaissances nécessaires pour traiter cette masse de données de façon automatique. Quelques expériences isolées ont pourtant montré l'intérêt que peut avoir la connaissance de l'activité du serveur pour ses utilisateurs [Pri99].

L'utilisation d'un mediaspace est inévitablement liée à la vie privée de ses utilisateurs. Un système basé sur le protocole HTTP peut être utilisé pour la communication audiovisuelle entre individus sans exposer leur vie privée. Il suffit pour cela de prendre en compte les utilisateurs qui se situent du côté du serveur et de leur permettre d'être conscient et d'agir sur l'activité du système. En recueillant les informations disponibles concernant les clients qui accèdent à leurs services, les serveurs peuvent offrir aux utilisateurs observés ou écoutés tous les éléments nécessaires aux mécanismes de contrôle et de notification décrits dans le chapitre précédent. En donnant ainsi aux utilisateurs un pouvoir de décision avant l'exécution du service demandé, il est possible d'inverser le rapport de vulnérabilité qu'introduisent la présence de caméras et de microphones dans leur environnement.

4. Résumé du chapitre

Les travaux sur les mediaspaces ont montré l'intérêt d'une approche de la communication entre individus centrée sur les usages et non sur la technologie. Le choix de l'infrastructure logicielle pour réaliser ce type d'environnement doit donc s'inscrire dans cette démarche et ne pas se faire sur des critères purement techniques.

Nous avons évoqué dans ce chapitre les avantages que peut apporter une conception modulaire de l'infrastructure logicielle d'un mediaspace. Nous avons expliqué que les plates-formes traditionnelles de développement de composant logiciels sont inadaptées à la réalisation d'un mediaspace parce que leurs mécanismes de conception, configuration et composition sont inaccessibles aux utilisateurs. Nous avons enfin proposé le Web comme alternative à ces plates-formes. Nous avons présenté les mécanismes de désignation, transmission et composition de ressources qui constituent la base du Web. Nous avons ensuite montré comment ces mécanismes peuvent être utilisés pour créer l'infrastructure logicielle et les interfaces d'un mediaspace.

Dans le chapitre suivant, nous présentons deux systèmes pour la communication entre individus que nous avons développés suivant l'approche décrite dans ce chapitre.

Chapitre IV

Mediascape et videoServer

Depuis 1993, le groupe Interaction Homme-Machine du Laboratoire de Recherche en Informatique de l'Université Paris-Sud dispose d'un réseau audio/vidéo analogique permettant de relier entre eux ses différents bureaux ainsi que divers équipements tels qu'un magnétoscope, des tuners TV ou des cartes d'acquisition audio et vidéo.

Au fil des années et des thésards du groupe, différents prototypes logiciels ont été développés pour établir des connexions, capturer et transmettre l'audio et la vidéo sous forme numérique, gérer des sessions collaboratives associées à ces moyens audiovisuels ou proposer différentes formes de contrôle et de notification. Ainsi en 1994, alors que le Web commençait à se peupler d'internautes, les utilisateurs de Mosaic qui consultaient les pages HTML du groupe pouvaient déjà y voir des images fixes des différents bureaux capturées en temps réel.

Suite à l'explosion du Web, l'accès simple, rapide et interactif à des sources d'information distantes est aujourd'hui aussi banal que l'échange de courrier électronique. Nous avons vu dans le chapitre précédent que les standards qui constituent réellement le Web peuvent servir de base à une infrastructure logicielle pour la communication médiatisée.

Dans ce chapitre, nous présentons deux systèmes que nous avons développés pour intégrer l'accès à un mediaspace dans l'environnement logiciel par une approche centrée sur les documents : *Mediascape* et *videoServer*.

1. Mediascape

Mediascape [Rou99b] est notre premier prototype de mediaspace développé pour explorer les utilisations du Web comme infrastructure logicielle pour la communication médiatisée. Mediascape utilise le réseau audio/vidéo analogique installé au LRI. La Figure 40 présente la configuration matérielle utilisée. Dans un souci de simplicité, la figure ne montre que deux nœuds et n'inclut pas l'équipement audio.

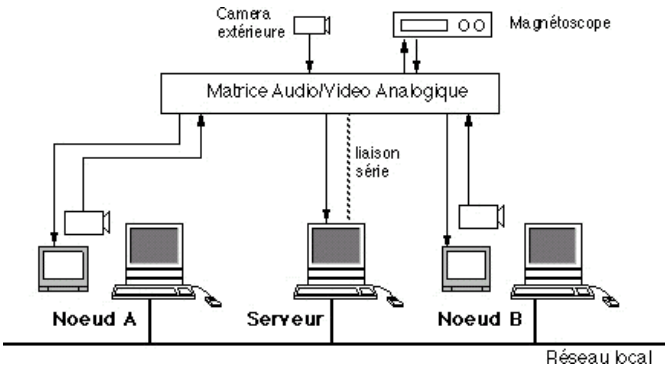


Figure 40. Configuration matérielle de Mediascape

Les connexions audio/vidéo sont établies par l'intermédiaire d'une matrice de commutation analogique pilotée par un serveur par liaison série. Le serveur est également équipé d'une carte d'acquisition connectée à la matrice qui permet de numériser des images provenant de n'importe quelle source vidéo du réseau. Les nœuds A et B sont des nœuds individuels typiques, similaires à ceux présentés dans le Chapitre II, avec une caméra, un moniteur et une station de travail. Une caméra extérieure filmant le campus et un magnétoscope sont également connectés à la matrice.

1.1. Implémentation de Mediascape

L'implémentation actuelle[23] de Mediascape repose sur un serveur HTTP spécialement développé pour la gestion des connexions audio/vidéo et la numérisation d'images. Ce serveur, implémenté en Python[24], offre les services suivants :

- /glance, un service de type coup d'œil utilisant une liaison vidéo bidirectionnelle de quelques secondes.
- /call, un service de type visiophone reposant sur une liaison audio/vidéo bidirectionnelle de durée indéterminée.

[23] Mediascape, tel qu'il est décrit dans cette section, fonctionne en permanence depuis juin 1997

[24] http://www.python.org/

- /register, qui permet à un utilisateur d'indiquer au système le nœud du mediaspace le plus proche de sa position actuelle dans les bâtiments.

- /authlevel, qui lui permet de choisir entre trois niveaux d'accessibilité : tous les services sont acceptés, le service `glance` seulement, ou aucun.

Les ressources utilisées par le serveur de Mediascape pour les services `glance` et `call` ne contiennent que le nom du service demandé et celui de la personne à contacter (`/call.nicolas` par exemple). L'identification de l'utilisateur à l'origine d'une requête se fait par l'utilisation d'un cookie. Lorsqu'une requête ne contient pas ce cookie, le serveur retourne un formulaire d'identification au client, considérant qu'il s'agit de la première utilisation du système. Ce formulaire demande à l'utilisateur d'indiquer son nom et de choisir dans une liste le nœud du mediaspace de plus proche de sa position (Figure 41). Ces informations sont stockées par le serveur de Mediascape qui renvoie au client un cookie contenant le nom de l'utilisateur qui devra être renvoyé avec chaque demande ultérieure. Les utilisateurs peuvent par la suite changer le nœud du mediaspace qui leur est associé en envoyant une commande `register`, telle que `/register.b224`.

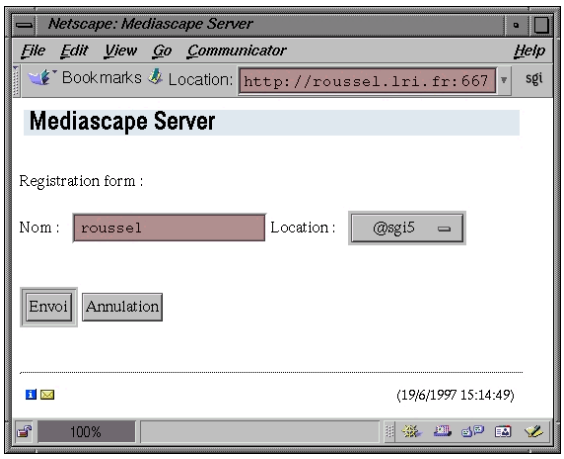


Figure 41. Formulaire d'identification de Mediascape

La Figure 42 illustre les trois étapes correspondant au traitement par Mediascape d'une demande de service audio/vidéo analogique :

1. Le navigateur Web envoie une requête HTTP au serveur de Mediascape. Cette requête précise le nom du service demandé ainsi que celui de la personne à contacter et est accompagnée d'un cookie qui permet d'identifier l'utilisateur.

2. Le serveur de Mediascape utilise deux tables d'indirection pour pouvoir passer des noms des deux utilisateurs concernés à leur dernière position connue puis aux

équipements audiovisuels présents à ces emplacements. Il envoie ensuite par la liaison série les commandes nécessaires à la matrice de commutation audio/vidéo pour relier entre eux ces différents équipements.

3. Le serveur de Mediascape renvoie une réponse HTTP au client lui indiquant que sa requête a été traitée et n'a généré aucune nouvelle information.

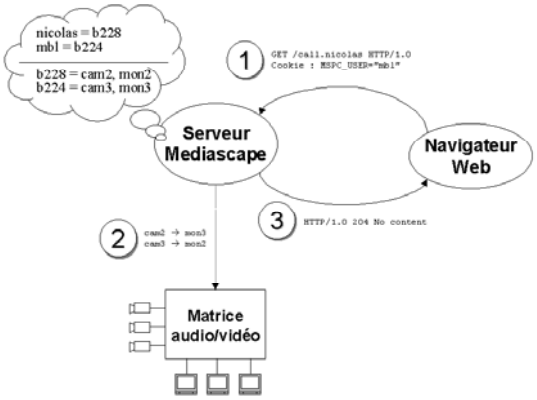


Figure 42. Traitement d'une requête `call` par le serveur de Mediascape.

En plus des services audio/vidéo analogiques accessibles uniquement aux membres de notre groupe, deux autres services sont accessibles de façon plus large :

- `/grab`, pour capturer (numériser) une image fixe de l'une des sources. Les images sont disponibles en plusieurs tailles, de 80x60 à 640x480 pixels et peuvent subir une correction gamma pour améliorer leur lisibilité. Les paramètres spécifiant la taille et la correction gamma sont passés par l'intermédiaire des options de la requête.

- `/postit`, pour permettre aux autres utilisateurs locaux et distants de laisser des messages sur l'écran informatique du correspondant.

La Figure 43 illustre les quatre étapes correspondant au traitement par Mediascape d'une demande de service `grab` :

1. Le client envoie une requête HTTP indiquant le nom de la personne recherché.

2. Le serveur de Mediascape utilise ses tables d'indirection pour passer du nom de la personne au nœud puis à la caméra la plus proche de cette personne. Il envoie ensuite par la liaison série les commandes nécessaires à la matrice de commutation audio/vidéo pour relier cette caméra à l'entrée vidéo de la machine sur laquelle il tourne.

3. Il numérise l'une des images arrivant sur l'entrée vidéo de la machine et la code au format JPEG.

4. Il envoie une réponse HTTP au client indiquant que la requête a été exécutée et à produit une image codée au format JPEG. Les données JPEG sont jointes à la réponse.

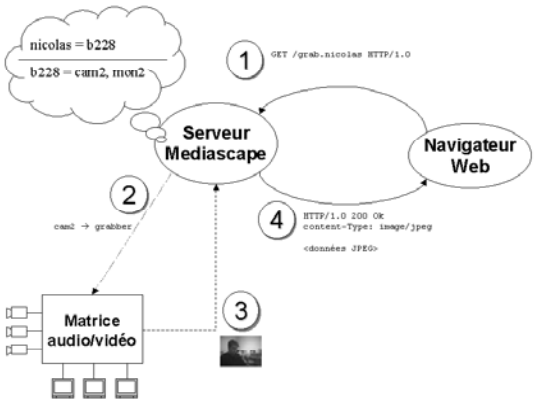


Figure 43. Traitement d'une requête `grab` par le serveur de Mediascape

La composition des PostIts se fait par un formulaire HTML envoyé par le serveur. Le message composé peut contenir du code HTML (Figure 44).

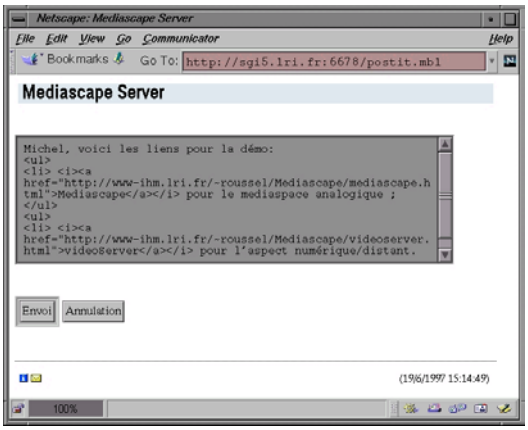


Figure 44. Composition d'un PostIt Mediascape dans un navigateur Web

Du code HTML est automatiquement ajouté au message pour y inclure le nom de l'expéditeur ainsi qu'une image prise par sa caméra qui pourra être utilisée pour le rappeler plus tard (Figure 45). La possibilité offerte par les navigateurs Netscape d'être commandés par une autre application [Net94] permet au serveur de Mediascape de demander au navigateur du correspondant d'afficher le message sur l'écran informatique le plus proche de sa dernière situation géographique connue.

Figure 45. Exemple de PostIt affiché. En demandant le réaffichage du message par le navigateur, le destinataire peut mettre à jour l'image de l'auteur du message. En cliquant sur celle-ci, il peut établir une liaison audio/vidéo avec lui.

Le serveur permet enfin de concaténer plusieurs demandes de service délimitées par / pour former une unique requête, permettant ainsi d'enchaîner plusieurs coups d'œil :

```
/glance.michel/glance.stephane/glance.mountaz
```

1.2. Intégration de Mediascape

HTML propose un certain nombre de balises qui permettent d'inclure dans un document des images (`<img ...>`), des liens vers d'autres documents ou services (`<a ...> … </<a>`) ou encore d'autres objets plus complexes (`<embed ...>`). Quelques simples lignes de HTML suffisent donc pour accéder aux services de Mediascape depuis n'importe quel navigateur Web. On peut par exemple :

- inclure une image d'un des bureaux, capturée au moment où le document est affiché

```
<img src="http://mediascape/grab.nicolas">
```

- ajouter à un texte des liens qui permettent de contacter les membres de notre groupe

```
<a href="http://mediascape/call.nicolas"> Nicolas </a>
et <a href="http://mediascape/call.stephane"> Stéphane </a>
connaissent le code de la salle de visioconférence
```

- combiner les deux exemples précédents en insérant une image capturée qui sert de bouton pour établir une connexion audio/vidéo

```
<a href="http://mediascape/call.nicolas">
   <img src="http://mediascape/grab.nicolas">
</a>
```

- ajouter du code JavaScript à un lien qui jette un coup d'œil lorsque la souris passe sur le lien

```
<a href="http://www-ihm.lri.fr/~roussel/"
onMouseOver="window.location='http://mediascape/glance.nicolas"
> N. Roussel </a>
```

- insérer un lien vers un formulaire de saisie de PostIt

```
<a href="/postit.paul"> Note pour Paul </a>
```

- insérer une commande qui exécute une suite de coups d'œil, comme si l'on traversait un couloir en regardant au passage à travers les portes

```
<embed src="http://mediascape/glance.michel/glance.stephane">
```

L'interface de base de Mediascape est un document HTML qui affiche des images des différents nœuds du mediaspace (Figure 46). Ces images sont éventuellement transformées par le serveur pour refléter l'accessibilité des utilisateurs. Selon un principe similaire à celui de CAVECAT, trois icônes représentant une porte ouverte, entrouverte et fermée sont utilisées pour choisir un des trois niveaux du service `authlevel`. Un utilisateur peut jeter un coup d'œil à un autre en déplaçant la souris au-dessus de son nom, et établir une connexion audio/vidéo en cliquant sur son image ; ces services utilisant le matériel analogique (moniteur T.V., caméra, etc.) et ne modifiant pas le document interface. Pour chaque utilisateur, deux icônes donnent également accès au service de PostIt et à une passerelle pour envoyer un courrier électronique.

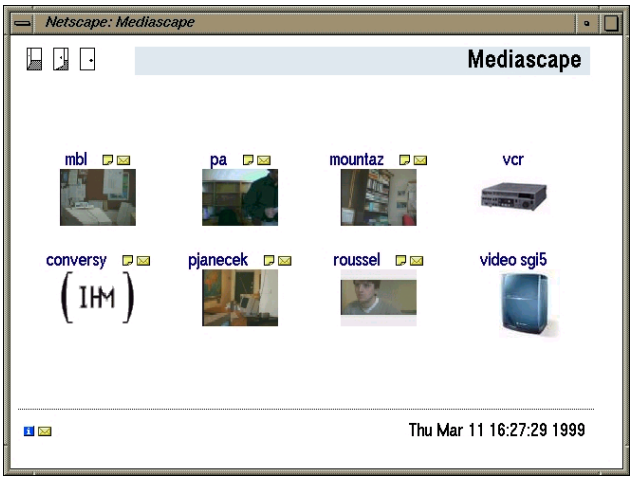


Figure 46. Interface de base de Mediascape. Conversy a fermé sa porte et celle de Roussel est entrouverte. Les deux icônes de droite donnent accès à des sources vidéo publiques, un magnétoscope et une station de travail

L'en-tête du document contient une instruction spéciale qui indique au navigateur qu'il doit recharger le document après quelques minutes pour le conserver à jour. Comme nous l'avons expliqué précédemment, l'utilisation de cookies par le serveur permet de simplifier le nommage des ressources pour utiliser par exemple `/call.nicolas` au lieu de `/X.call.nicolas` et rend donc possible l'utilisation du même document par tous les utilisateurs.

1.3. Flexibilité de Mediascape

L'interface de base de Mediascape étant un document HTML comme les autres, les utilisateurs habitués à ce type de langage peuvent en faire une copie et la modifier pour créer leurs propres interfaces. Ils peuvent par exemple choisir de ne montrer qu'un sous-ensemble des nœuds, de supprimer des icônes pour réduire la taille de l'interface ou au contraire en ajouter pour contrôler une caméra distante. Ce type d'adaptation est décrit par R. Bentley et P. Dourish comme une personnalisation de surface[25] [BD95] : les utilisateurs peuvent choisir parmi un nombre prédéfini d'options, déterminées par les possibilités offertes par HTML et les autres langages disponibles dans les navigateurs, tels que JavaScript ou VRML.

Jusqu'à présent, nous n'avons vu que des documents HTML dédiés au mediaspace. Le code HTML permettant d'accéder aux services de Mediascape peut aussi être ajouté à des documents existants ou nouveaux dont le but premier n'est pas de servir d'interface au mediaspace. Plusieurs personnes collaborant à un projet peuvent insérer des images ou des commandes de Mediascape dans les documents relatifs au projet. Ceci permet à un membre quelconque du groupe de savoir rapidement qui est là et de pouvoir engager des conversations lorsqu'il visite la page Web du projet ou travaille sur un document particulier. Ces fonctionnalités de coordination intégrées dans les documents peuvent être très utiles en cas de rédaction à plusieurs auteurs d'un article. En plus des images fixes des différents auteurs (Figure 47), une commande peut être ajoutée pour les connecter automatiquement lorsque l'un d'eux regarde le document.

[25] traduction de *surface customization*

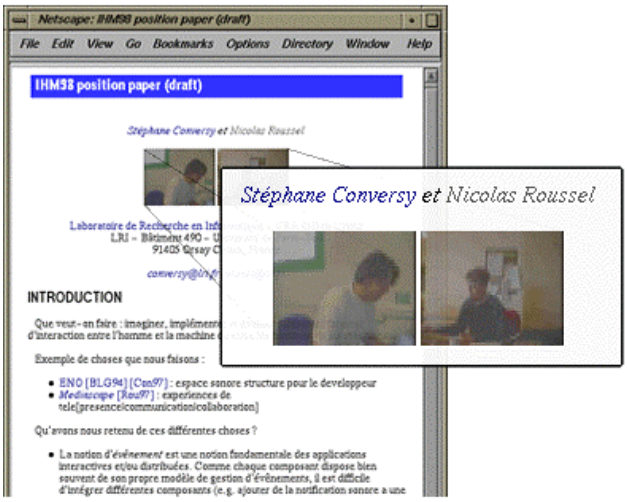


Figure 47. Coordination entre deux coauteurs par un document augmenté par Mediascape. Ici, nous pouvons voir que les deux auteurs sont présents en même temps

Un nombre croissant de personnes lisant leur courrier électronique avec des applications qui comprennent HTML et HTTP, comme les navigateurs Web, il est possible d'inclure des commandes Mediascape dans les messages qui seront exécutées à leur lecture, fournissant une nouvelle fois des moyens de coordination implicite entre l'expéditeur et le destinataire (Figure 48).

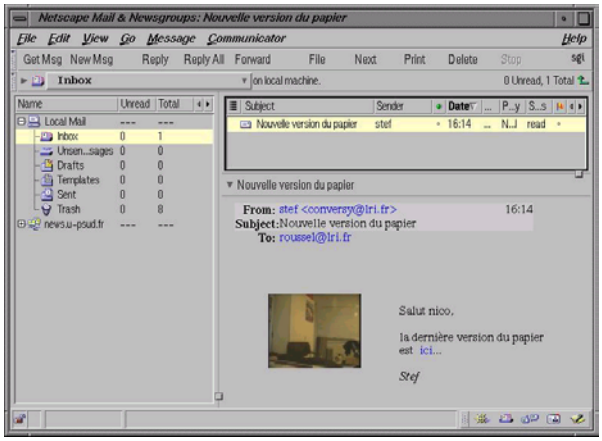


Figure 48. Intégration des services de Mediascape dans un outil de messagerie électronique. L'image affichée est capturée au moment de la lecture du message.

1.4. Intimité de Mediascape

Comme nous l'avons indiqué, Mediascape permet à ses utilisateurs de choisir parmi trois niveaux d'accessibilité. Le serveur consigne également toutes les requêtes dans un fichier. Les utilisateurs peuvent consulter ce fichier pour savoir si quelqu'un les a appelés pendant leur absence. Ils peuvent également utiliser des applications qui surveillent pour eux ces fichiers et déclenchent

diverses notifications lorsqu'une personne demande une connexion ou qu'une image est capturée par le serveur.

Tandis que les services analogiques `glance` et `call` sont limités à une communauté restreinte d'utilisateurs locaux, le service de capture d'images est accessible depuis la page Web de notre groupe[26] et est souvent utilisé par des personnes en dehors de notre laboratoire. Pour le contrôle d'accès, ces demandes d'images numérisées sont donc considérées comme des demandes de coup d'œil. Tandis que les services analogiques sont basés sur une règle de réciprocité stricte, la capture d'image repose sur une version relâchée de cette règle : les personnes distantes peuvent nous voir tandis que nous ne pouvons pas les voir, mais nous pouvons par contre savoir d'où elles viennent et éventuellement qui elles sont grâce aux informations consignées dans le fichier de log par le serveur.

2. VideoServer

Mediascape offre un accès local à notre matériel audio/vidéo analogique et un accès distant à des images fixes provenant de ce matériel et capturées à la demande. La plupart des machines de notre groupe pouvant être équipées de caméras numériques, nous avons décidé d'étendre notre mediaspace en lui ajoutant un composant logiciel dédié à la transmission de flux vidéo numériques et qui puisse nous permettre de communiquer avec des collègues, amis ou parents éloignés.

Nous avons conçu et implémenté videoServer [Rou99a], une application qui tourne sur la machine personnelle d'un individu et lui permet de rendre accessible par HTTP des images fixes ou animées, enregistrées ou capturées en temps réel. Contrairement aux serveurs HTTP habituels destinés à des groupes de personnes, entreprises, laboratoires ou associations, videoServer est un serveur destiné à un usage strictement personnel. Il est conçu pour être le point d'accès unique à l'image de son propriétaire.

VideoServer est écrit en C++ et tourne efficacement sur nos machines UNIX (SGI IRIX ou Linux[27]). Il utilise le mécanisme de server-push décrit dans le Chapitre III pour transmettre par HTTP des flux vidéo décrits sous forme de séquences d'images JPEG. HTTP, ou plus exactement TCP, n'offre aucune garantie de qualité de service. Le nombre d'images transmises par seconde dépend donc de la bande passante disponible au moment de la transmission. Une image JPEG de 160x120 pixels montrant une

[26] http://www-ihm.lri.fr/

[27] Voir l'Annexe A pour plus de précisions sur la capture de vidéo en temps réel sur ces plates-formes.

personne à son poste de travail faisant environ 3 Ko, le mécanisme de server-push nous permet toutefois de transmettre des flux vidéo de cette dimension à 15 images par seconde ou plus suivant l'état du réseau.

2.1. Intégration de videoServer

Certains navigateurs étant capables d'afficher une série d'images transmises par server push à la place d'une simple image, videoServer met la vidéo à la portée de toute personne ayant accès à Internet.

Afin de pouvoir être utilisé sur une machine sur laquelle tourne déjà un serveur Web classique, videoServer utilise le port TCP `5555`, différent du port HTTP standard (`80`). Une image fixe, une séquence vidéo en temps réel ou une séquence préenregistrée peuvent être incluses dans un document HTML à la place d'une image classique en utilisant le code suivant :

```
<img src="http://roussel.lri.fr:5555/push/photo">
<img src="http://roussel.lri.fr:5555/push/video">
<img src="http://roussel.lri.fr:5555/push/movie/aLaPlage">
```

Le dernier exemple demande au videoServer de transmettre le fichier préenregistré nommé "aLaPlage". Une autre application, videoRecorder, permet d'enregistrer des séquences vidéo dans le format du serveur.

2.2. Flexibilité de videoServer

Pour les images capturées en temps réel, plusieurs options peuvent être ajoutées à la requête pour indiquer si le flux vidéo peut être préempté par une autre application – voir l'Annexe A – ou pour spécifier l'entrée vidéo à utiliser, caméra numérique ou entrée analogique par exemple, ainsi que le taux de compression et le facteur de changement d'échelle. Ces deux derniers paramètres permettent aux utilisateurs d'adapter le service fourni par videoServer à la bande passante habituellement constatée entre deux sites en réduisant la résolution ou en augmentant le taux de compression. Pour les séquences vidéo, deux autres paramètres permettent également de spécifier le nombre d'images à transmettre ainsi que le temps de pause entre deux images successives. En faisant varier ces deux valeurs il est possible, en plus de la liaison vidéo classique, de créer de nouveaux services de type coup d'œil ou vue d'ensemble en ne transmettant que quelques images seulement ou en choisissant un temps de pause de quelques minutes :

```
/push/video?node=camera&zoom=8&cratio=20.0&locked=1&length=20
```

Les utilisateurs peuvent se servir de code HTML plus sophistiqué que les exemples précédents. JavaScript permet ainsi d'afficher une image fixe qui se transforme en flux vidéo lorsque le curseur de la souris est situé au-dessus, et qui retourne à son état figé lorsque le curseur en sort :

```
<a
href=http://www-ihm.lri.fr/
onMouseOver='document.img1.src="http://videoserver/push/video"'
onMouseOut='document.img1.src="http://videoserver/push/photo"'>
<img name="img1" src="http://videoserver/push/photo" >
</a>
```

Nous utilisons ce type de code pour créer des documents montrant une vue d'ensemble à partir de plusieurs videoServers (Figure 49).

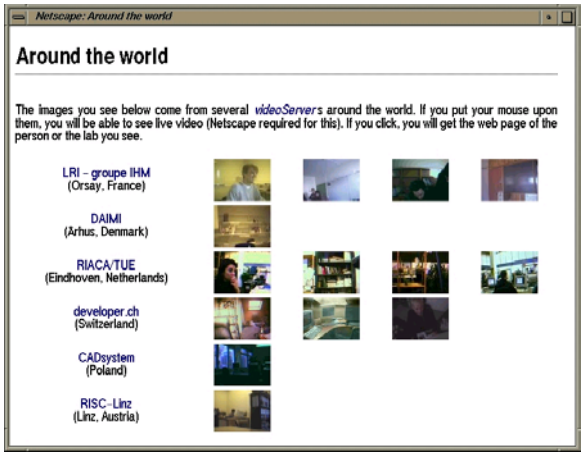


Figure 49. Document HTML montrant plusieurs videoServers. Les images sont fixes par défaut, mais se transforment en flux vidéo au passage de la souris.

Associé au modèle en couches de HTML 4.0, JavaScript permet également de superposer un flux vidéo au-dessus du document affiché dans un navigateur (Figure 50).

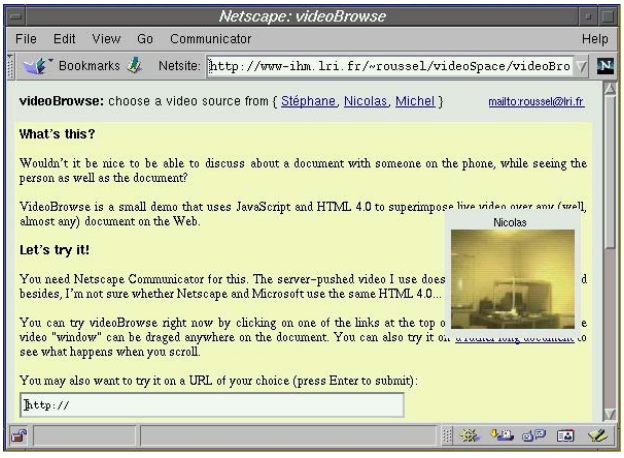


Figure 50. Utilisation de JavaScript pour ajouter une "fenêtre vidéo" flottante sur un document HTML

Il est ainsi possible de discuter d'un document avec une personne au téléphone tout en voyant la dite personne dans la même fenêtre que le document.

2.3. Intimité de videoServer

Pour chaque requête qu'il reçoit, videoServer exécute un programme externe de notification et de contrôle : le notifier. VideoServer transmet au notifier un certain nombre d'informations concernant la requête : le numéro du processus qui va la traiter, le nom d'utilisateur de la personne distante s'il le connaît, le nom ou l'adresse IP de la machine distante ainsi que l'URL du document HTML qui a mené le client au serveur et qui est généralement spécifiée dans le champ `Referer` de la requête. Il lui transmet également une description du type de service demandé (photo, vidéo ou fichier préenregistré) ainsi que des paramètres éventuels (zoom, qualité, etc.). Le notifier doit retourner la description du service à exécuter, qui peut être différent de celui originellement demandé. VideoServer prévient également le notifier de la fin de l'exécution du service.

Le notifier par défaut est un script Shell UNIX qui permet au propriétaire de videoServer de définir simplement différentes politiques de contrôle d'accès à son image et d'être conscient de son utilisation. Différentes formes de notifications peuvent ainsi refléter le service demandé ou l'identité de la personne distante par l'affichage de messages sur l'écran de la station informatique ou l'utilisation d'icônes auditives. Le notifier pouvant exécuter n'importe quelle commande avant, pendant ou après l'exécution du service, il est également possible d'ajouter une liaison audio à l'aide d'une autre application telle que SpeakFreely[28] ou de déclencher une connexion vidéo vers la personne distante en envoyant une commande à un navigateur Web sur la machine locale.

Le notifier pouvant modifier le service à exécuter, il lui est également possible de changer la résolution ou le taux de compression des images, leur nombre ou le délai entre deux images successives. Un taux de compression élevé permet par exemple de transmettre des images avec une qualité dégradée, suffisante pour savoir combien de personnes sont présentes, mais insuffisante pour les identifier formellement (Figure 51). En changeant le taux de compression en fonction de l'identité de la personne distante, il est donc possible de filtrer simplement l'information transmise par les images.

[28] http://www.speakfreely.org/

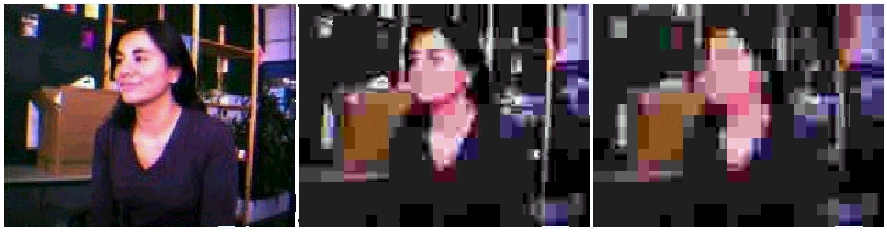


Figure 51. Dégradation de la qualité de l'image par augmentation du taux de compression JPEG

Le service demandé peut être modifié comme dans l'exemple que nous venons de voir, mais il peut également être complètement redéfini. Il est ainsi possible de transmettre une séquence vidéo préenregistrée, quelques images montrant un bureau vide par exemple, au lieu d'une séquence capturée en temps réel. Grâce au mécanisme de redirection de HTTP, il est aussi possible de diriger le client vers une autre ressource, éventuellement située sur un autre serveur.

Lorsqu'une demande de vidéo en temps réel ne précise pas le nombre d'images à transmettre, videoServer limite ce nombre à 5000, ce qui correspond au plus à trois minutes. Ceci garanti à l'individu qui a lancé le serveur qu'il ne peut être surveillé de façon permanente sans que cette surveillance soit périodiquement remise en cause et éventuellement renégociée. Ces demandes de connexion répétées sont rapidement remarquées lorsque chaque service est associé à une notification auditive, produisant le même effet qu'une personne qui se tiendrait devant la porte ouverte d'une maison, le doigt appuyé sur la sonnette. Connaissant le nom de la machine à l'origine de la demande et éventuellement le nom d'utilisateur de la personne distante, il est généralement facile de faire comprendre à cette personne que l'on est conscient de sa présence par des gestes, un message sur une feuille de papier ou un courrier électronique.

VideoServer traite chaque requête indépendamment des autres et crée pour cela un nouveau processus système. Un numéro permettant d'identifier ce processus est passé au notifier en plus des informations mentionnées précédemment. Le notifier ou toute autre application exécutée par lui, peut utiliser ce numéro pour stopper le processus en lui envoyant un signal. Cette possibilité nous a permis de développer une application qui représente les connexions vidéo actives par des éléments graphiques affichés sur l'écran informatique, le propriétaire de videoServer pouvant agir sur ces éléments pour obtenir des détails les concernant, le nom de l'utilisateur distant par exemple, ou stopper leur exécution.

3. Usages de videoServer

La première version de videoServer a été rendue publique en juin 1998 et différentes versions de celui-ci sont ou ont été utilisées par différents collègues, amis ou parfaits inconnus. Poussés par des motivations diverses, certains l'ont essayé, d'autres l'ont adopté, et quelques-uns uns ont bien voulu témoigner. Dans cette section, nous présentons une synthèse des usages de videoServer observés au long des deux dernières années.

3.1. Conscience mutuelle et coordination

VideoServer est décrit par l'un de ses premiers propriétaires et utilisateurs comme "the advanced finger tool".

Finger [Zim91] est l'un des plus anciens services permettant d'obtenir des informations sur les utilisateurs d'un système informatique. Ce service repose sur un couple client/serveur. Différents "capteurs" informatiques (le clavier, l'accès au système de fichiers, etc.) permettent au serveur de savoir depuis combien de temps une personne n'a pas utilisé sa machine ou si elle a du courrier électronique en attente. Le client permet aux utilisateurs d'interroger le serveur sur l'activité d'une ou plusieurs personnes. En plus de ces informations sur l'utilisation du système informatique, finger permet à chaque utilisateur de placer dans un fichier nommé `.plan` des renseignements le concernant qui seront affichées par les clients. Ce fichier sert généralement à indiquer ses coordonnées ou ses projets à court ou moyen terme, tels que ses déplacements prévus ou son emploi du temps.

Comme finger, videoServer est effectivement souvent utilisé pour renseigner sur l'activité d'une ou plusieurs personnes. A la manière du fichier `.plan`, l'utilisation du Web comme interface permet d'associer aux images de videoServer du texte ou des liens vers d'autres documents, ajoutant ainsi à l'information immédiate – la présence ou l'absence de la personne à l'image – des informations utiles pour une coordination sur le long terme.

La consultation des fichiers de log de videoServer permet à son propriétaire de savoir qui a cherché à le contacter pendant son absence, chose qui est impossible à savoir avec finger. Mais nous allons voir que la différence entre finger et videoServer est beaucoup plus fondamentale et tient surtout à deux approches très différentes : l'un perçoit l'activité tandis que l'autre permet de la percevoir.

3.2. Une bonne image vaut mieux qu'un long discours

Notre cerveau est particulièrement bien adapté à l'analyse d'images. Il réduit chaque seconde l'équivalent de plusieurs millions de bits d'information arrivant du système visuel à quelques dizaines de bits

perçus de façon consciente [Nør91]. En quelques fractions de seconde, nous pouvons ainsi interpréter les images de la Figure 52 en termes de présence, d'activité et de disponibilité des personnes. Cette capacité de filtrage des images s'accompagne d'une impressionnante capacité de concentration qui nous permet de passer rapidement d'une perception périphérique à une exploration plus détaillée des informations perçues[29]. En regardant de plus près une image, nous pouvons par exemple y apercevoir des détails nous renseignant de façon plus précise sur l'activité d'un individu. Enfin, un grand nombre de facteurs sociaux-culturels nous aident à interpréter ces images. Notre interprétation des images de la Figure 52 repose ainsi largement sur notre connaissance de l'habitude de placer les caméras au-dessus des moniteurs.

Figure 52. Exemples d'images typiques de videoServer

Comme d'autres systèmes plus récents tels que *PeepHoles* [Gre96] ou *AROMA* [PS97], finger présente une vue synthétique abstraite de l'activité des individus construite à partir de différents "capteurs" informatiques. Cette analyse par la machine présente certains avantages, comme la possibilité de renseigner sur l'activité passée des personnes. Mais malgré l'amélioration des techniques de détection et de suivi par analyse d'images, aucun système n'est aujourd'hui capable de maintenir une description des positions, identités et rôles des objets ou acteurs de son environnement [CCB00].

S'il est vrai qu'une bonne image vaut mieux qu'un long discours, elle vaut également mieux que ce que les informations qui peuvent en être extraites par les systèmes informatiques actuels. L'accès aux services de videoServer par des documents HTML permet de rendre aisément accessible l'image des membres d'un groupe, offrant ainsi un moyen simple et efficace de juger de l'activité et de la disponibilité de chacun. Comme nous l'avons vu, les utilisateurs peuvent composer leurs propres vues d'ensemble sur les personnes

[29] [Nør91] propose une expérience très simple pour illustrer la différence entre ce que nous percevons et ce que nous ressentons de façon consciente. Fermez les yeux et essayez d'identifier les différentes sources sonores qui vous entourent. Vous n'aviez sans doute pas conscience de l'existence de toutes ces sources auparavant. Il vous a pourtant suffit de quelques instants et d'un peu d'attention pour les percevoir.

qui les intéressent plus particulièrement. Contrairement à des systèmes comme Polyscope [BT91] ou Portholes [DB92], cette notion de groupe d'intérêt n'est pas définie une fois pour toutes de façon centralisée, mais peut être librement définie par chaque utilisateur pour chaque document HTML utilisé comme vue d'ensemble.

J. Tang et al. soulignent à juste titre l'importance de la rapidité des services numériques de type coup d'œil ou vue d'ensemble [TR94]. Les vues d'ensemble utilisant plusieurs videoServers présentent des images capturées au moment de l'accès au document. Les temps d'acquisition et de transmission s'additionnant, le chargement du document peut donc prendre de une à plusieurs secondes suivant la taille des images demandées. En conséquence, les vues d'ensemble réalisées avec videoServer utilisent généralement des images de très petite taille, de l'ordre de 80x60 pixels. Mais jusqu'à quelle taille l'image reste-t-elle réellement utile ?

3.3. Une (très) petite image vaut-elle encore quelque chose ?

[JG99] présente les résultats d'une étude expérimentale visant à isoler le type d'information utilisé à partir d'images similaires à celles de la Figure 52 pour juger de la disponibilité des personnes observées. L'un des buts de cette étude était en particulier de déterminer la résolution minimale nécessaire pour véhiculer les informations utiles à la conscience mutuelle. L'étude reposait sur quatre séries d'images montrant un homme ou une femme dans dix situations différentes, comme entrant ou sortant de la pièce, travaillant, ou utilisant le téléphone. Chaque série était disponible dans six différentes tailles (16x16, 32x32, 64x64, 128x128, 256x256, et 512x512). Une image de chaque résolution était présentée aux sujets de l'expérience qui devaient alors noter la disponibilité de la personne sur une échelle de 1 (le moins disponible) à 7 (le plus disponible).

Cette étude de Johnson et Greenberg conclue à l'existence d'un seuil au-dessous duquel il devient difficile d'interpréter les images. Ce seuil se situe, selon eux, entre 64x64 et 128x128 pixels. Les implications pratiques de cette conclusion sont les suivantes : les auteurs conseillent l'utilisation d'images de résolution au moins égale à 128x128 pour toute application ou étude utilisant des images fixes pour déterminer l'activité d'un groupe de personnes.

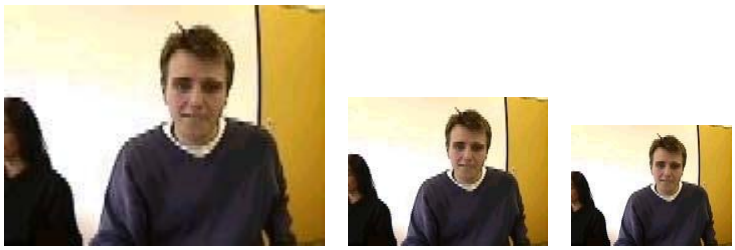


Figure 53. La même image, présentée à des résolutions de 128x96, 80x60 et 64x48

Les images 80x60 utilisées pour les vues d'ensemble de videoServers se situent bien au niveau du seuil identifié par Johnson et Greenberg. A proportions équivalentes, il y a cependant une différence importante entre une image à une résolution de 128x96 et la même image à une résolution de 80x60 (Figure 53). Comment peut-on alors expliquer le fait qu'aucun des utilisateurs de videoServer n'ait jugé cette taille trop petite ? L'explication vient, selon nous, des nouvelles possibilités offertes par le navigateur Web pour l'exploration des images. La méthode d'exploration habituelle pour lever un doute lors de l'interprétation d'une image consiste à la regarder plus attentivement. Cette exploration nécessite donc un bon niveau de détail. L'utilisation du navigateur Web et de videoServer propose deux nouvelles méthodes : le rafraîchissement de la vue et l'utilisation de la vidéo.

Le réaffichage du document HTML servant de vue d'ensemble provoque la mise à jour des images provenant des différents videoServers. D'un simple clic de souris, il est donc possible de demander une nouvelle vue qui peut apporter des informations supplémentaires, permettant par exemple de savoir si la personne en haut à droite de la Figure 52 vient d'arriver ou s'en va. Nous avons montré dans la section précédente un exemple d'utilisation de JavaScript qui permet de transformer une image fixe en flux vidéo lorsque la souris passe au-dessus. Les vues d'ensemble utilisant ce type de code permettent donc d'obtenir rapidement des détails sur une image particulière. Une succession d'images affichées peut ainsi permettre d'identifier une personne que l'on distinguait mal sur l'image fixe (Figure 54).

Figure 54. Succession d'images 80x60

Nous venons de le voir, l'intégration des images fixes ou flux vidéo de videoServer dans les documents HTML offre de nouveaux moyens pour l'interprétation des images d'une vue d'ensemble en terme d'activité. D'autres vues, plus détaillées ou spécifiques, peuvent également être reliées aux vues d'ensemble par des liens hypertextes. Il est ainsi possible, en cliquant sur l'une des images,

de voir apparaître une vue comme celle de la Figure 55 qui permet le contrôle de l'orientation et de la focale de la caméra distante et le changement de résolution de l'image affichée.

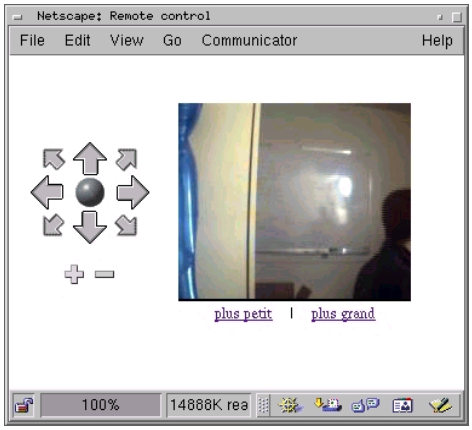


Figure 55. Contrôle d'une caméra par le Web : les icônes permettent de changer l'orientation et le zoom de la caméra, les liens sous l'image permettent de changer sa résolution

VideoServer est particulièrement utile lorsqu'une personne cherche à en contacter une autre, par exemple avant de téléphoner ou de se rendre dans son bureau. Les vues d'ensemble permettent de savoir si la personne est là et, si c'est le cas, le passage à l'utilisation de la vidéo ou le changement pour une vue plus détaillée et modifiable permettent de se faire une idée assez précise de son activité et de sa disponibilité.

3.4. Mécanismes de notification et communication gestuelle

Plusieurs propriétaires de videoServer s'en servent pour rester en contact avec des membres de leur famille éloignés. En intégrant un flux d'images de videoServer dans une page Web, ils peuvent être accessibles de façon permanente et nécessitant peux de connaissances et aucun logiciel particulier pour celui qui veut les regarder[30].

Les mécanismes de notification de videoServer transforment ce qui aurait pu être un nouveau Big Brother en un instrument de dialogue. L'un des comportements les plus typiques observé chez les propriétaires de videoServer est le petit signe amical généralement envoyé à celui qui regarde. Chez certains, ce geste est même devenu machinal, se déclenchant dès la notification sonore et précédant souvent l'affichage à l'écran de l'origine de la requête. Ce geste, s'il se veut amical, est aussi un acte de défense envers les inconnus : il prévient la personne distante que l'on est conscient de

[30] Comparez la simplicité d'une URL face aux explications sur l'installation et le fonctionnement d'un outil de visioconférence classique pour une personne agée qui veut voir ses petits-enfants...

sa présence. Lorsque cette personne est identifiée comme amicale, par une notification sonore particulière par exemple, d'autres modifications du comportement peuvent être utilisées - sourire, déplacement du corps, exagération des expressions - pour lui faire comprendre que sa présence a été signalée (Figure 56).

Figure 56. Avant et après la notification : communication mimo-gestuelle répondant à la notification du videoServer

Avec le temps et une utilisation régulière avec un groupe de personnes connues (collègues, famille, amis, etc.), un certain côté ludique finit par s'installer qui, associé à l'absence de liaison audio, favorise une forme de communication mimo-gestuelle entre le propriétaire de videoServer et ses observateurs. Les images permettent alors d'exprimer des choses simples comme "on se téléphone plus tard", "je t'ai envoyé un mail", "je m'en vais" ou "qui est dans le bureau ?". Une liaison vidéo bidirectionnelle facilite bien évidemment cette communication, puisqu'elle permet de vérifier si le message a bien été compris.

En l'absence de liaison bidirectionnelle, la personne distante ne dispose que d'un seul moyen de communication immédiat : le rechargement du document, qui provoque l'établissement d'une nouvelle connexion au videoServer et déclenche à nouveau les mécanismes de notification. Ce moyen de communication, bien que très primaire, se révèle toutefois très utile puisqu'il permet d'attirer facilement l'attention.

3.5. L'image en renfort du texte ou de la voix

VideoServer est régulièrement utilisé comme moyen de communication additionnel lors d'échanges parlés ou textuels. La simplicité de mise en œuvre est généralement ce qui motive cette utilisation. La qualité des images transmise est parfois également citée par les personnes habituées à d'autres logiciels dédiés à la communication vidéo. Ces logiciels étant conçus principalement dans un souci d'utilisation minimale du réseau, ils ont en effet recours à une quantification des couleurs et à la compression inter-images qui peuvent donner un côté artificiel au flux vidéo affiché. Le fond peut par exemple se transformer en une série de blocs de pixels homogènes visiblement inchangés au fil des images.

Plusieurs propriétaires de videoServer ont développé des applications de dialogue textuel qu'ils utilisent avec celui-ci pour pouvoir communiquer les personnes qui les regardent (Figure 57). Ces applications ont été implémentées comme des applets Java, ce

qui permet d'y accéder depuis un navigateur Web, et servent aussi bien à discuter en famille qu'à des entretiens d'embauche à distance.

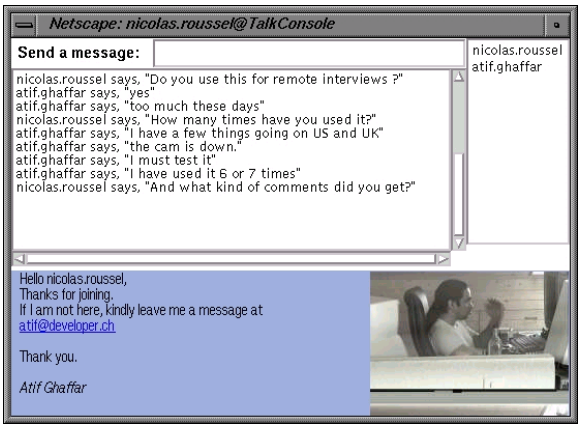


Figure 57. Utilisation de videoServer avec une applet Java de dialogue textuel

Comme nous l'avons indiqué au début de cette section, videoServer est très utile pour décider du moment d'un appel téléphonique. La liaison vidéo est très souvent conservée pendant la durée de la conversation, permettant d'interpréter les silences ou bruits éventuels et d'associer les autres personnes éventuellement présentes à la conversation. VideoServer a ainsi été utilisé par un professeur en visite sabbatique en Europe pour assister à une soutenance de l'une de ses étudiantes de Master du Canada. La liaison vidéo était accompagnée d'une liaison téléphonique. Selon les termes des personnes concernées par cet essai, "la qualité de l'image était plutôt bonne et le taux de rafraîchissement excellent". Bien que l'image ait été trop peu précise pour permettre la lecture à distance, la liaison vidéo a été jugée particulièrement utile pour synchroniser les changements de transparents. Elle a aussi permis à la personne distante d'avoir une vue d'ensemble de la salle au moment des questions.

4. Notification, contrôle et adaptation du contenu

La plupart des serveurs HTTP permettent de refuser l'accès d'un client à une ressource ou de le rediriger vers une autre ressource. Les mécanismes de contrôle utilisés pour cela reposent sur l'identification de la personne distante par la machine qu'elle utilise ou la saisie d'un mot de passe. Comme nous l'avons fait remarquer au chapitre précédent, les serveurs HTTP sont généralement utilisés de manière collective et tournent souvent sur une machine indépendante et dédiée. Cette situation de service collectif et d'isolement explique en partie l'absence des mécanismes de notification qui permettraient aux personnes qui mettent de l'information en ligne d'être prévenues lors de l'accès à cette information.

Lorsque nous mettons une version préliminaire d'un article, un rapport de fin de contrat ou des photos de vacances sur le Web, ces informations sont accessibles par les personnes concernées aussi bien que par de parfaits inconnus. La mise en œuvre des moyens de contrôle d'accès des serveurs HTTP étant souvent complexe, nous acceptons généralement cette surexposition de l'information, en espérant naïvement que l'accès sera limité aux personnes à qui nous avons donné l'URL correspondante. La plupart des propriétaires de videoServer gèrent l'accès à leur image de cette même manière. Les images capturées en temps réel par les caméras placées dans leur bureau sont constamment accessibles à tout le monde. Toutefois, les mécanismes de notification et de contrôle du notifier associé au videoServer permettent de raffiner cette politique en fonction des utilisations observées.

4.1. Utilisation des mécanismes de notification et de contrôle de videoServer

Le notifier de videoServer permet d'associer des notifications sonores ou visuelles aux différentes combinaisons de services et d'utilisateurs distants. Une fois mis en place, ces mécanismes de notification préviennent le propriétaire chaque fois qu'une personne le regarde. Ainsi, lorsqu'une personne se connecte plusieurs fois de suite au videoServer, les notifications répétées finissent généralement par attirer l'attention de son propriétaire. Celui-ci peut alors modifier les règles de contrôle d'accès pour traiter différemment les requêtes venant de cette personne, en se basant sur son nom ou celui de sa machine.

Les modifications successives du notifier permettent de passer progressivement d'un accès public ouvert à tous à un accès filtré puis éventuellement restreint à quelques utilisateurs. Accompagnant cette évolution, les possibilités de redéfinition des services de videoServer permettent d'exécuter une version dégradée d'un service plutôt que de l'interdire. En réduisant la taille des images transmises, en changeant leur qualité de compression ou en limitant leur nombre ou leur fréquence, le propriétaire du serveur peut contrôler la quantité d'information transmise.

L'un des paramètres passés par videoServer au notifier est l'URL de la ressource qui pointait vers le serveur. En plus des informations concernant l'identité de la personne distante et le service demandé, les notifications peuvent donc indiquer le contexte dans lequel à eu lieu cette requête. Ainsi, lorsque notre téléphone sonne quelques instants après qu'une personne ait accédé à notre serveur depuis un document ou figurent également nos coordonnées, nous pouvons

raisonnablement supposer qu'il s'agit de la même personne et décrocher en l'appelant par son nom[31].

L'intégration d'images de videoServer dans des documents HTML permet de profiter des mécanismes de notification du notifier pour être prévenu de l'accès à ces documents. Cette expérience nous a amenés à penser que la généralisation de mécanismes de notification et d'adaptation du contenu similaires à ceux proposés par videoServer permettrait des formes de communication plus subtiles que celles permises par les serveurs et clients HTTP actuels pour des documents HTML classiques. Nous avons réalisé un prototype de serveur de documents reposant sur ces deux concepts.

4.2. MyWebServer

MyWebServer est un serveur HTTP destiné à rendre accessible un ensemble de documents hypertexte. Contrairement aux autres serveurs HTTP, myWebServer n'est pas destiné à un groupe ou à une organisation, mais à un unique individu. Il offre à son propriétaire un ensemble de moyens lui permettant de contrôler l'accès aux documents, d'adapter leur contenu en fonction de l'identité de la personne qui les consulte, et d'y associer d'éventuelles notifications.

Chaque requête reçue par myWebServer déclenche une recherche de l'identité de la personne distante par plusieurs moyens : Ident daemon [Joh85] distant, table de correspondance associant utilisateurs et noms de machines et, en dernier ressort, identification par mot de passe. Le document demandé est ensuite produit à partir d'un fichier contenant des macro-commandes. Ces fichiers peuvent être vus comme des programmes dont l'exécution dépend de la personne qui les demande et dont le résultat sera transmis au client distant.

Les macro-commandes sont utilisées pour produire le code HTML du document et peuvent également servir à restreindre l'accès à certaines parties ou à déclencher différentes notifications. Elles sont décrites en Python et peuvent être chargées dynamiquement dans le serveur à partir de feuilles de style, des fichiers contenant du code Python, ou même être incluses dans la description du document auquel elles s'appliquent (Figure 58).

```
#import html

#inline {
   import string
```

[31] A ce jour, l'expérience n'a encore jamais ratée et toujours fait impression, malgré la banalisation des identificateurs d'appels téléphoniques...

```
   def process_list(items):
      output = "<UL>"
      for i in string.split(items):
         output = "%s\n<LI>%s"%(output,i)
      output = "%s\n</UL>"%output
      return output
}

#header {Choses à acheter…}

#list {pommes poires scoubidous}

#footer
```

Figure 58. Exemple de description de document pour myWebServer. Le document utilise la feuille de style `html` et définit une nouvelle macro (`list`).

Nous utilisons myWebServer pour rédiger tous nos documents depuis environ deux ans. Au fur et à mesure de son utilisation, nous avons écrit un certain nombre de macro-commandes qui nous permettent de disposer aujourd'hui d'un système de production de documents HTML assez complet, se chargeant par exemple de la gestion des références bibliographiques et de la numérotation des figures. MyWebServer nous a servi pour la rédaction d'articles scientifiques, de sujets de travaux dirigés pour l'enseignement et de documentation pour différents projets de recherche. Nous l'avons également utilisé pour rédiger la version initiale de cette thèse qui a ensuite été éditée avec un logiciel plus adapté à la mise en page et à l'édition sur support papier.

Dans la suite de cette section, nous donnons deux exemples significatifs illustrant l'utilité des possibilités d'adaptation du contenu et de notification de myWebServer pour des activités quasi-quotidiennes.

4.3. Adaptation du contenu de supports pour l'enseignement

Une feuille d'exercices de travaux dirigés peut être vue comme composée de trois parties. La première partie contient les énoncés, la seconde contient les notes utiles à l'enseignant pour la conduite de la séance et la troisième contient les corrections des exercices. Si les énoncés doivent être toujours lisibles par tous, il est généralement souhaitable que la correction ne soit disponible qu'à la fin de la séance et que les notes de l'enseignant lui soient strictement réservées.

Les serveurs HTTP traditionnels gèrent le contrôle d'accès au niveau du système de fichier et ne peuvent accéder à une granularité plus fine, comme une partie de document. Leur utilisation impose donc le découpage d'une feuille d'exercices en trois fichiers distincts et la mise en œuvre de mécanismes de contrôle d'accès différents sur chacun d'entre eux. Ce découpage

forcé présente plusieurs inconvénients. Tout d'abord, il oblige l'enseignant à séparer l'énoncé, la correction et les notes concernant un même exercice alors que ces éléments sont intrinsèquement liés. Cette séparation obligatoire le conduit en général à créer des redondances entre les éléments pour que chacun soit compréhensible individuellement. D'autre part, le contrôle d'accès étant défini au niveau du système de fichier, le renommage ou le déplacement de l'un des trois fichiers oblige l'enseignant à modifier les mécanismes de contrôle associés.

MyWebServer permet de modifier le contenu d'un document en fonction de la personne qui le demande. Il est donc possible, en déclarant des identités distinctes telles que "étudiant en fin de séance" et enseignant, de restreindre l'accès à certaines parties d'un document comme les corrections ou notes pour l'enseignant. Un unique fichier peut donc être utilisé pour la feuille d'exercices. Le contrôle d'accès de myWebServer étant décrit à l'intérieur du document et évalué au moment de l'exécution des macro-commandes, l'unique fichier décrivant la feuille d'exercices peut être déplacé ou renommé librement.

```
#section {Exercice 1}

Donnez les différentes étapes nécessaires pour tracer un
polygone
en projection perspective avec Z-buffer.

#restrict enseignant {
  Rappel de l'algorithme du Z-buffer :
     …
}

#restrict {enseignant correction} {
  Z ne peut pas être interpolé linéairement à cause de la
  projection perspective : il faut se rammener à une vue
  parallèle
  …
}
```

Figure 59. Exemple de description pour myWebServer d'un énoncé avec notes pour l'enseignant et correction

La Figure 59 montre un exemple d'énoncé accompagné de notes pour l'enseignant et d'une correction. Suivant le rôle de la personne qui le consulte (public, correction ou enseignant), ce fichier peut produire trois documents différents comportant plus ou moins d'information, tout en restant accessible par la même URL.

4.4. Coordination par la notification pour des communications scientifiques

La communication scientifique par le Web pose deux problèmes, l'un lié à la confidentialité des informations échangées, l'autre à la

difficile coordination entre les personnes distantes. La rédaction d'un article scientifique ou d'une thèse passe généralement par une succession de phases d'écriture par une ou plusieurs personnes puis de lecture pour évaluation. Le bon déroulement de ces alternances de phases peut devenir critique lorsque les personnes impliquées sont très éloignées et disposent de peu d'occasions de communiquer. Il est alors important de pouvoir savoir quand une personne accède ou a accédé au document.

La Figure 60 montre un exemple de description d'article scientifique pour myWebServer. Seul le titre et le résumé sont publics, le reste du document n'est accessible qu'à deux utilisateurs particuliers, `nicolas` et `michel`. Trois notifications sont incluses dans la partie à accès restreint. La première est une icône auditive destinée à refléter l'identité de la personne qui consulte le document. La deuxième notification affiche l'identité de cette personne ainsi que le nom du document à l'écran. La troisième enregistre ces deux informations dans un fichier pour garder une trace au cas où le propriétaire de myWebServer ne soit pas présent pour entendre ou voir les deux premières notifications.

```
#import {article biblio controlNotif}

#header {Support informatique à une communication médiatisée}

#author #authorIsMe

#frAbstract { Nos travaux portent sur … }

#frKeywords {Collecticiel, Mediaspace, Vidéo, Web}

#restrict {nicolas michel} {
  #notify audio client
  #notify screen {client document}
  #notify log {client document}

  #section INTRODUCTION

  #para { Les mediaspaces #cite{wendy-mediaspaces} sont des
  environnements intégrant audio, vidéo et informatique pour
  … }

  …

  #biblio Bibliographie BiblioTeX unsrt
}

#footer
```

Figure 60. Exemple de description d'un article scientifique incluant des mécanismes de contrôle d'accès et de notification

4.5. Perspectives pour l'évolution de la communication par le Web

Les serveurs Web sont perçus comme de simples mécaniques de transmission de données. Ce sont pourtant de véritables outils de communication médiatisée. La conception de ces systèmes devrait donc être centrée sur l'utilisateur et son contexte, et non uniquement sur des problèmes de performances et de réseau. MyWebServer diffère des serveurs HTTP traditionnels par les possibilités de notification et d'adaptation de l'information transmise. Nous l'avons vu, ces possibilités modifient la façon dont le Web peut être utilisé pour la communication entre individus en simplifiant la production de documents et facilitant la coordination.

Les études d'utilisabilité du Web ont jusqu'à présent principalement porté sur les clients, c'est-à-dire les navigateurs, et les langages de description de document utilisés. VideoServer et myWebServer montrent, selon nous, qu'il est nécessaire de s'intéresser aux serveurs HTTP eux-mêmes, afin de proposer des systèmes qui permettent de communiquer par le Web autrement qu'en mettant le bon fichier au bon endroit et en s'envoyant des courriers électroniques pour s'assurer que tout s'est passé comme prévu.

5. Résumé du chapitre

Nous avons présenté dans ce chapitre une première approche de l'intégration de la vidéo dans les systèmes informatiques qui repose sur l'utilisation du Web pour intégrer de façon transparente les services de communication dans des documents. Nous avons illustré cette approche par Mediascape et videoServer, deux serveurs HTTP qui permettent de contrôler une infrastructure audio/vidéo analogique et de transmettre des flux vidéo numériques.

Nous avons montré comment Mediascape permet d'intégrer les services d'un mediaspace analogique dans des documents HTML, le rendant ainsi rapidement accessible, utilisable de façon intuitive et facilement personnalisable. Nous avons ensuite montré comment videoServer nous permet d'étendre le mediaspace analogique à des sites distants par l'intermédiaire du mécanisme de server push et l'utilisation de documents HTML pour transmettre et afficher des flux vidéo numériques. Nous avons présenté les mécanismes offerts par videoServer pour la notification et le contrôle d'accès et montré comment ils permettent à son propriétaire de conjuguer accessibilité et vie privée.

La transmission d'images JPEG par server push utilisée par videoServer peut sembler simpliste aux spécialistes du codage et de la transmission informatique de flux vidéo. Cependant, comme nous l'avons plusieurs fois répété, notre approche de la vidéo n'est

pas focalisée sur les techniques mises en œuvre elles même, mais sur les usages qu'elles permettent. Sur ce point, nous avons présenté dans ce chapitre plusieurs usages de videoServer qui le différencient très nettement des logiciels de communication vidéo traditionnels et le rapprochent des environnements de type mediaspace.

Les mécanismes de notification, de contrôle et d'adaptation du contenu de videoServer proposent une nouvelle vision du Web dans laquelle les serveurs, comme les clients, sont conçus pour les utilisateurs et ne sont plus de simples diffuseurs de données. Nous avons montré que ces mécanismes peuvent être appliqués à des échanges de données autres que des images à travers la présentation de myWebServer, un serveur de documents conçu spécifiquement pour un usage personnel.

Dans le chapitre suivant, nous présentons plusieurs applications clientes et protocoles développés pour videoServer qui nous permettent de passer progressivement de l'usage des navigateurs Web que nous venons de présenter à d'autres applications.

Chapitre V

Au-delà du navigateur : nouveaux clients et nouveaux protocoles

Dans le chapitre précédent, nous avons vu comment le Web peut être utilisé comme infrastructure pour construire un environnement vidéo destiné à la coordination et à la communication au sein d'un groupe. Nous avons montré comment Mediascape et videoServer utilisent HTTP pour commander un mediaspace analogique et transmettre de la vidéo numérique à distance et comment les navigateurs Web peuvent servir d'interface à ces deux systèmes. Nous avons vu comment cette approche permet de prototyper rapidement de nouveaux services en copiant/collant des portions de code HTML ou en s'échangeant les documents correspondants par courrier électronique. Dans ce chapitre, nous montrons comment il est possible d'étendre les possibilités offertes par cet environnement en remplaçant les navigateurs Web standards par des applications et protocoles dédiés.

1. VideoClient

L'utilisation de navigateurs Web pour afficher les images venant de videoServers a vite montré ses limites. Cette solution occupe beaucoup de place sur l'écran et en mémoire, et les possibilités d'interaction sont très limitées. Il est ainsi difficile de redimensionner les images une fois la liaison établie. Afin de pouvoir conserver une vue sur nos collègues ou amis indépendamment de l'utilisation du navigateur, nous avons donc réalisé une application indépendante, *videoClient*, destinée à afficher un flux vidéo dans une fenêtre X Window.

VideoClient a été conçu initialement pour afficher des images ou séries d'images JPEG transmises par HTTP. Cette application permet donc d'afficher les images et flux vidéo transmis par videoServer, mais également les captures de Mediascape ou les images des nombreuses caméras accessibles sur le Web par server-push[32] (Figure 61).

Figure 61. VideoClient

L'URL de la source vidéo à afficher est passée en argument lors du lancement de videoClient :

```
videoClient -i http://neapolis.daimi.au.dk:5555/push/video
```

Lorsque la fenêtre de l'application est redimensionnée, les images affichées sont redimensionnées, agrandies ou rétrécies, pour occuper tout l'espace disponible. Cette possibilité permet à l'utilisateur d'adapter la taille des flux vidéo affichés sur son écran en fonction du contexte de leur utilisation. Un flux peut être affiché en petit dans un coin jusqu'à ce que l'utilisateur note la présence de la personne distante et décide de lui téléphoner. Il peut alors déplacer et agrandir la fenêtre vidéo pour mieux voir son interlocuteur.

1.1. VideoClient et le Web

Comme nous l'avons vu au Chapitre III, les données échangées sur le Web sont typées selon la classification MIME [FB96]. La plupart des navigateurs Web permettent aux utilisateurs d'associer à un type MIME une application externe[33] qui sera exécutées pour traiter les données du type spécifié à la place du navigateur. Nous avons utilisé cette possibilité pour rendre videoClient accessible depuis le navigateur Netscape. Pour cela, nous avons créé un nouveau type MIME expérimental : le type `video/x-videoSpace` (Figure 62).

[32] Voir par exemple le site http://www.allcam.com/ pour de nombreux exemples

[33] Ces applications externes sont aussi appelées *helper applications*

```
Decription: videoSpace stream
MIME Type: video/x-videoSpace
Suffixes: .vs
Application: /home/roussel/CodeBuild/videoSpace/web/helper %s
```

Figure 62. Description du type MIME `video/x-videoSpace` pour son utilisation par le navigateur Netscape

L'application externe associée à notre nouveau type de données est un script Shell UNIX qui est chargé de lancer videoClient, les données reçues par le navigateur décrivant la source à afficher (Figure 63).

```
#!/bin/sh

FILE=$1
URL=`cat $FILE`

exec $CODEBUILD/videoSpace/videoApps/videoClient -i $URL
```

Figure 63. Exemple de script de liaison entre le navigateur et videoClient

Une fois le type `video/x-videoSpace` décrit au navigateur, il existe deux moyens pour déclencher l'exécution de videoClient par celui-ci. La première possibilité consiste à modifier la configuration du ou des serveurs HTTP utilisés pour que les fichiers dont l'extension est `.vs` soient envoyés avec le type MIME `video/x-videoSpace`. Un tel fichier contient alors l'URL de la source vidéo à afficher.

La deuxième possibilité repose sur l'utilisation d'un schéma particulier d'URL, le schéma `data:` [Mas98]. Ce schéma permet d'insérer dans un document HTML une référence à des données qui sont elles-mêmes codées dans le document. Il est ainsi possible d'inclure un lien déclenchant l'exécution de videoClient pour une source vidéo donnée :

```
<a
href="data:video/x-videoSpace,http://sgi5.lri.fr:5555/push/video">
   Nicolas Roussel (videoClient)
</a>
```

Ce type de lien entre le navigateur et videoClient permet d'ajouter une dimension supplémentaire aux interfaces réalisées en HTML et JavaScript décrites au chapitre précédent : il est possible d'afficher des images fixes de bureaux distants dans le navigateur qui se transforment en vidéo au passage de la souris et qui déclenchent l'exécution de videoClients lorsque l'utilisateur clique dessus. Il est donc possible de passer d'une vue d'ensemble statique à un flux d'image pour un individu puis à une communication bidirectionnelle dédiée, le tout en un clic de souris.

1.2. VideoClient et les autres applications

Une fenêtre X Window est une zone a priori rectangulaire dans laquelle une application peut dessiner. A une application peut correspondre un nombre quelconque de fenêtres, sans aucune contrainte sur leur organisation spatiale à l'écran. Le serveur X maintient une structure arborescente décrivant des liens de parenté entre les fenêtres qui est utilisée pour la distribution et la propagation des événements vers les applications. Un gestionnaire de fenêtres est généralement utilisé, qui décore les fenêtres de plus haut niveau de l'arbre et permet de les redimensionner, de les déplacer ou de les iconifier. Ce que l'utilisateur perçoit comme étant une fenêtre est donc en réalité un ensemble de fenêtres X Window qu'il peut manipuler (Figure 64).

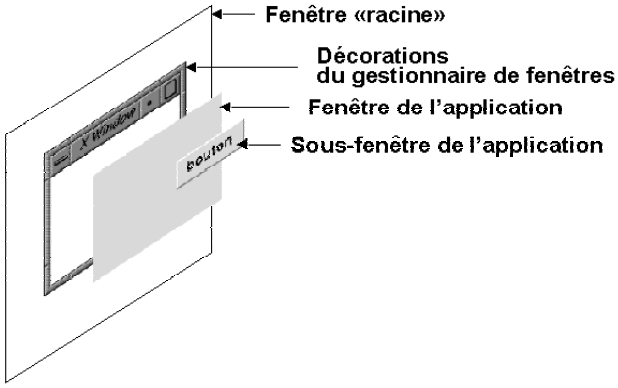


Figure 64. Principe du fenêtrage X Window. La fenêtre «racine» correspond au fond de l'écran. Ce que l'utilisateur perçoit comme étant l'application est en fait composé de trois fenêtres : les décorations, la fenêtre principale et le bouton.

En créant la fenêtre de videoClient en un point particulier de l'arborescence X Window plutôt qu'à la racine, il est possible de soustraire l'affichage des flux vidéo du gestionnaire de fenêtres et de l'intégrer au sein d'autres applications. Au lieu d'être une fenêtre indépendante et manipulable par l'utilisateur, videoClient permet dans ce cas d'"augmenter" les applications existantes. La Figure 65 montre une application de type *talk*, qui permet à deux utilisateurs de dialoguer en tapant du texte, associée à un videoClient montrant l'un des deux participants.

Figure 65. l'application talk augmentée par videoClient

Dans cet exemple, un script Shell UNIX permet d'obtenir l'identificateur de la fenêtre talk et de créer la fenêtre vidéo comme fenêtre fille de celle-ci (Figure 66). Pour l'utilisateur, ces deux fenêtres sont indissociables : elles sont déplacées, icônifiées et passent ensemble de l'avant à l'arrière plan.

```
#!/bin/sh
# Usage: vtalk user video-url

PEER=$1
URL=$2

TITLE="vtalk-$PEER"

# launch an xterm with the talk application
xterm -T $TITLE -n $TITLE -e talk $PEER &

# wait for the window to be mapped
sleep 5

# get the window ID and launch the videoClient
WID=`xwininfo -name $TITLE | grep "Window id" | awk 'print $4'`

DST="glxwindow://localhost:0/?parent="$WID"&geometry=120x90-
120"

exec videoClient -i $URL -o $DST
```

Figure 66. Script Shell UNIX réalisant le talk augmenté

Certaines boîtes à outils proposent des éléments d'interface explicitement conçus pour héberger des applications externes. Ainsi, les cadres (*frames*) de la boîte à outils Tcl/Tk[34] peuvent être déclarés container, ce qui signifie que leur contenu est géré par une autre application. La Figure 67 montre un exemple de code Tcl qui crée un tel cadre dans lequel est placé un videoClient.

```
# create a frame widget for the video
frame .tv -container true -width 160 -height 120
```

[34]http://www.scriptics.com/

```
set id [winfo id .tv]

# launch a videoClient with this frame as parent
set pid [exec videoClient \
     -i http://roussel.lri.fr:5555/push/video \
     -o "glxwindow://localhost:0/?parent=$id" &]

# make the widget visible
pack .tv
```

Figure 67. Code Tcl permettant d'insérer un videoClient dans un container Tk

La Figure 68 montre un exemple d'application basée sur ce code qui permet d'afficher une source d'images choisie parmi plusieurs à l'aide de boutons.

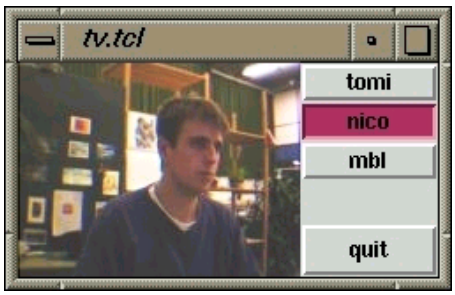


Figure 68. Intégration de videoClient dans une application Tcl/Tk

L'accès à videoClient depuis un langage comme Tcl/Tk permet de prototyper rapidement et simplement des applications autonomes destinées à la coordination, la communication ou la collaboration à distance. Nous avons réalisé une application qui permet à un utilisateur de conserver en permanence sur son écran une vue présentant plusieurs les images de plusieurs sources d'images, telles que Mediascape, des videoServers ou des Webcams. L'application dispose d'un mode spécial dans lequel l'affichage se limite à une simple barre qui révèle les images au passage du pointeur de la souris (Figure 69). De même que l'écriture de code HTML, la programmation en Tcl/Tk est à la portée de nombreux utilisateurs. Ceux ci peuvent aisément modifier le code de l'application pour modifier la liste des sources d'images à afficher ou ajouter des fonctionnalités, comme l'envoi de courrier électronique ou la commande du réseau analogique par Mediascape.

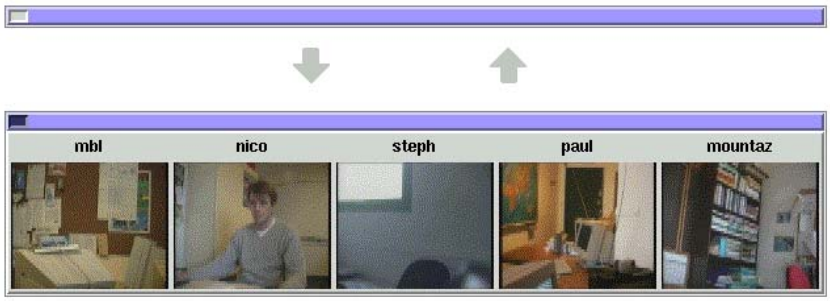


Figure 69. Intégration de plusieurs videoClient dans une application Tcl/Tk de type vue d'ensemble

La possibilité d'insérer videoClient dans un cadre Tk permet également d'"augmenter" des applications Tcl/Tk existantes en changeant quelque peu leur code. Les liaisons vidéo peuvent ainsi avantageusement compléter les autres formes de communication proposées par l'application d'origine, comme le dialogue textuel, et permettent de réduire le travail de coordination nécessaire à l'accomplissement des tâches de production. Nous avons utilisé cette approche avec GroupKit [RG96], une boîte à outils basée sur Tcl/Tk pour le développement d'applications collaboratives.

La Figure 70 montre *groupdraw*, une application de dessin partagé fournie avec GroupKit que nous avons modifiée pour ajouter des liaisons vidéo entre les utilisateurs. Les modifications effectuées utilisent les mécanismes de gestion d'entrée/sortie de participants de GroupKit pour associer un videoClient à chaque utilisateur distant. Ces modifications ajoutent 50 lignes à l'application originale qui en fait 500.

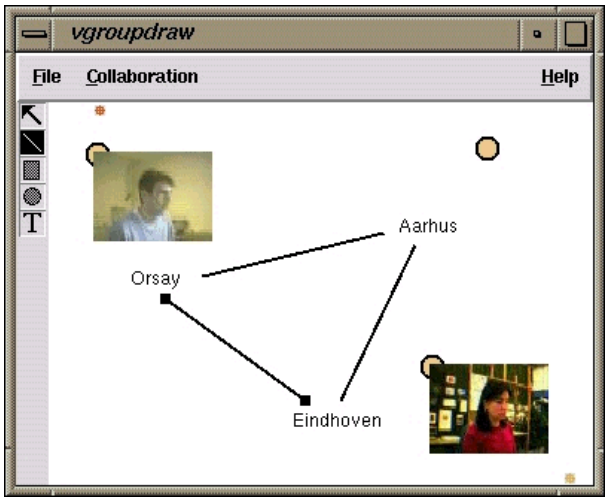


Figure 70. L'application groupdraw "augmentée" par l'ajout de liaisons vidéo entre les participants

Chaque utilisateur est représenté par un cadre contenant un videoClient, grâce à la propriété container décrite précédemment, et une poignée ovale qui sert à déplacer la vidéo sur la surface de

dessin. La poignée sans cadre vidéo indique à l'utilisateur où se trouve son image sur l'écran des autres participants. Les positions des différents cadres vidéo sont partagées par tous les participants : lorsque l'un d'entre eux déplace un cadre, il se déplace également sur les écrans des autres participants. Un participant peut donc entraîner un ou plusieurs autres utilisateurs vers une zone particulière de l'espace partagé, ce qui complète les mécanismes de conscience de groupe offerts en standard par GroupKit, tels que les télépointeurs ou les vues radar.

1.3. Du numérique à l'analogique

L'une des principales différences entre les mediaspaces numériques et les mediaspaces analogiques est la séparation entre l'écran informatique et l'écran vidéo rendue possible par ces derniers. Cette séparation est importante, non seulement parce qu'elle permet d'économiser de la place sur l'écran informatique, mais également parce qu'elle permet d'utiliser les capacités naturelles de perception périphérique pour la communication d'arrière-plan.

Certaines stations de travail de notre équipe étant capables de produire une sortie vidéo analogique, nous avons créé une variante de videoClient, nommée *analogVideoClient*, qui affiche les images sur un écran analogique (moniteur T.V.) au lieu de l'écran informatique. Nous avons également développé une application Tcl/Tk qui permet de faire passer un flux vidéo d'un écran à l'autre. Cette application utilise un videoClient pour afficher le flux vidéo dans une fenêtre de l'écran informatique. Lorsque cette fenêtre est déplacée vers un côté de l'écran, la vidéo disparaît, la fenêtre se transforme en onglet fixé au bord de l'écran, et la vidéo réapparaît sur l'écran analogique grâce à analogVideoClient. L'onglet permet ensuite de ramener le flux vidéo sur l'écran informatique.

2. Nouveaux protocoles pour videoServer

Internet a souvent été désigné comme "le réseau TCP/IP". Ces deux protocoles sont en effet à la base de la plupart des services qui composent Internet tel que nous le connaissons aujourd'hui. Comme nous l'avons vu au Chapitre III, HTTP [FGM+99] est basé sur TCP/IP. La transmission d'images par server-push HTTP se fait donc en réalité sur des connexions TCP/IP.

2.1. IP, TCP et les autres protocoles de la famille Internet

IP (*Internet Protocol* [Pos81a]) est un protocole d'interconnexion chargé de la transmission d'informations brutes entre deux points du réseau, indépendamment de la signification de ces données et de l'usage qui en est fait. IP transmet les données sous forme de paquets qui sont acheminés individuellement. C'est précisément ce découpage en paquets et les algorithmes de routage qu'il nécessite

qui caractérisent Internet et le différencie des autres technologies de réseaux.

TCP (*Transmission Control Protocol* [Pos81b]) est un protocole implémenté au-dessus de IP qui fournit un service de transport connecté fiable en mode flot : il découpe le flot de données émis en paquets, les transmet par IP et garantit que toutes les données émises sont reçues correctement par le destinataire et dans l'ordre de leur expédition. Pour assurer la fiabilité de la transmission, TCP assure la détection des pertes de paquets, contrôle l'intégrité des données reçues et effectue les éventuelles retransmissions nécessaires.

La famille des protocoles Internet comporte un autre protocole de niveau transport nommé *UDP* (*User Datagram Protocol* [Pos80]). A la différence de TCP, UDP transmet les données en mode non connecté et de façon non fiable : les datagrammes peuvent être perdus ou dupliqués, ou encore arriver dans le désordre. En outre, il permet de diffuser un unique datagramme à un groupe en un seul envoi[35]. TCP garantissant la réception correcte, sans perte et dans l'ordre des données émises, il est naturellement à la base de la plupart protocoles Internet que nous utilisons quotidiennement : le transfert de fichiers (*FTP* [PR85]), l'échange de messages électroniques (*SMTP* [Pos82] et *NNTP* [KL86]) ou l'accès à distance à une machine (*Telnet* [PR83]) se font à travers des connexion TCP.

2.2. Problèmes de congestion

La transmission de données par IP fonctionne selon le principe du "meilleur effort possible" (ou "best effort") : chaque nœud intermédiaire du réseau fait ce qu'il peut, traitant tous les paquets de la même manière. Lorsque la quantité de données à transmettre dépasse la capacité d'un des nœuds, on observe une augmentation du délai de bout en bout, de la gigue[36] et du taux de perte de paquets.

En cas de congestion du réseau, les retransmissions et le réordonnancement effectués par TCP peuvent contribuer à l'augmentation du délai de bout en bout et de la gigue, augmentant donc les variations du débit observé. Si ces variations sont rarement gênantes pour l'utilisateur lorsqu'il effectue un transfert de fichier ou envoie un courrier, elles deviennent extrêmement perturbantes dans le cas d'applications interactives. Ainsi, la variation du délai entre la frappe d'un caractère et son affichage peut transformer l'utilisation d'un programme d'accès distant comme telnet en une expérience terriblement frustrante.

---

[35] On parle alors de transmission UDP multicast

[36] La gigue est la variation du temps entre l'arrivée de deux paquets

Maintien d'un débit constant

Une solution possible pour limiter la variation du débit serait de permettre aux applications d'affecter des priorités aux paquets transmis pour refléter les contraintes temporelles qui pèsent sur leur utilisation. En cas de trafic important, il serait ainsi possible de mettre en attente certains paquets, comme le courrier électronique, pour laisser passer des paquets plus urgents contenant par exemple de l'audio ou de la vidéo. Il serait également possible de réserver certaines ressources réseau aux paquets les plus urgents. Malheureusement, contrairement à d'autres technologies réseau comme ATM, IP n'intègre pas ces notions de priorité, de réservation de ressources ou de qualité de service[37].

TCP, UDP et IP ne proposant pas de moyen de régulation du débit, ce problème doit être traité au niveau de l'application. Une méthode couramment utilisée consiste à stocker les données reçues dans un tampon et retarder leur utilisation. En préservant ce retard du traitement sur la réception, il peut ainsi s'assurer d'un débit quasi-constant. Cette solution est particulièrement bien adaptée aux services de "media à la demande" qui permettent de diffuser des séquences préenregistrées, des bandes annonces de films par exemple, ou des émissions de radio ou de télévision en direct vers un large public passif[38]. En effet, dans ce cas, l'application peut retarder autant qu'elle le veut l'utilisation des données, allant dans le cas extrême jusqu'à les télécharger complètement avant de les utiliser. Cette méthode est celle utilisée par des applications telles que RealPlayer, Quicktime player ou Windows Media *Player* pour accéder à des contenus audio et vidéo sur Internet.

Dans le cas d'échanges audio/vidéo multipoint, l'application ne peut retarder les flux reçus au-delà d'un certain seuil sans nuire à la qualité de l'interaction entre les participants. [IT97] situe ce seuil autour de 400 ms pour la communication audio, précisant qu'il est même préférable de perdre la synchronisation avec les images si cela permet de réduire le délai.

Adaptation du débit

En l'absence de garantie de qualité de service par les couches basses du réseau Internet, les contraintes temporelles liées au besoin d'interactivité sont incompatibles avec l'idée de débit constant. Afin de ne pas surcharger le réseau inutilement, les applications qui veulent transmettre de l'audio ou de la vidéo sur Internet devraient être capables d'adapter leur débit en fonction de l'état du réseau [DHT95]. TCP n'offre pas la souplesse nécessaire à

[37] Du moins, pour le moment. Pour avoir un aperçu des futures évolutions d'IP, voir [Coc00]

[38] "Passif" dans le sens ou il ne transmet rien...

cette adaptation. Par exemple, l'application n'est pas informée des pertes de paquets et n'a aucun moyen de contrôle sur la stratégie de retransmission. Les applications doivent donc se tourner vers d'autres protocoles.

Depuis quelques années, l'Internet Engineering Task Force développe *RTP* (*Real-time Transport Protocol* [SCFJ96]), un protocole dédié à la transmission en temps réel de flux audio et vidéo. RTP est destiné aussi bien aux applications de type "media à la demande" qu'aux applications interactives comme la téléphonie sur Internet. Bien qu'indépendant du protocole de transport utilisé, il est généralement implémenté au-dessus d'UDP.

RTP découpe les données à transmettre en paquets, et associe à chaque paquet un numéro de séquence et une estampille temporelle. L'estampille temporelle indique l'instant "couvert" par un paquet. Cette information permet de replacer les données reçues dans l'ordre. Le numéro de séquence, incrémenté à chaque paquet transmis, sert à détecter les pertes : les paquets composant une image ont la même estampille, mais ont des numéros de séquence différents.

La spécification de RTP décrit également RTCP (Real-Time Control Protocol). Ce protocole permet entre autres de synchroniser plusieurs transmissions par RTP, par exemple un flux audio et un flux vidéo. Il permet également à une application d'obtenir des informations concernant la qualité de la transmission des données. Cette information est obtenue par l'intermédiaire de rapports d'émission/réception régulièrement échangés entre émetteurs et récepteurs en plus des données initiales. Un certain nombre d'outils de recherche dédié à la communication audio ou vidéo ont été mis au point à partir de RTP [Tur95, Veg96] et utilisent ces rapports pour transmettre moins d'information ou réduire la fréquence d'émission.

2.3. VideoServer Transport Protocol

La variation de débit du réseau sur une transmission d'images de videoServer par server-push HTTP est rapidement perceptible : l'affichage se fige pendant l'attente de nouveaux paquets ou la retransmission de ceux perdus, puis la série d'images reconstituée défile comme en accéléré. Nous avons donc cherché une solution pour éviter ce problème.

La possibilité de spécifier dans la requête HTTP le temps entre deux images successives et le taux de compression JPEG permet de limiter la fréquence d'émission et la taille des données transmises par videoServer, ce qui a généralement pour conséquence de diminuer les variations de débit à la réception. Cependant, comme nous l'avons vu au chapitre précédent, l'un des intérêts de l'accès à videoServer par le Web est l'absence de phase d'initialisation,

d'appel ou de réglages qui favorise les échanges spontanés et imprévus. L'ajustement manuel par l'utilisateur du débit des données transmises par server-push en fonction de l'état courant du réseau n'est donc pas une solution satisfaisante. Nous avons donc ajouté un nouveau protocole à videoServer, accessible par videoClient et qui permet de limiter les variations de débit à la réception.

Notre nouveau protocole a été baptisé VSTP, pour *VideoServer Transport Protocol*. VSTP utilise une connexion HTTP pour transmettre la requête du client vers le serveur, et une liaison UDP pour transmettre les images du serveur vers le client (Figure 71).

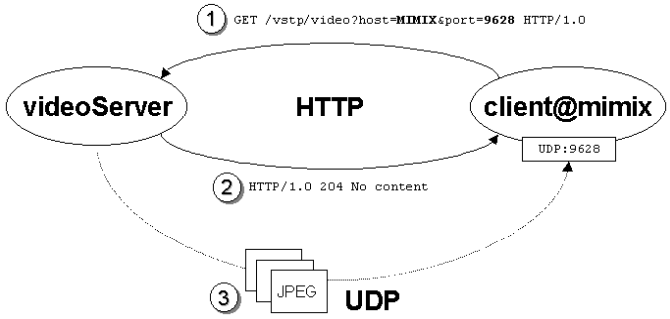


Figure 71. Utilisation de HTTP et UDP dans le cadre d'une liaison par VSTP : la requête est transmise par HTTP, les images par UDP

Les liaisons VSTP sont établies avec un videoServer par l'intermédiaire de requêtes HTTP, deux paramètres `host` et `port` permettant au client de donner au serveur le nom ou l'adresse de la machine et le numéro de port UDP nécessaires à l'envoi des images par ce protocole :

```
/vstp/photo?host=<STRING>&port=<INT>

/vstp/video?host=<STRING>&port=<INT>

/vstp/movie/path?host=<STRING>&port=<INT>
```

Chaque image à transmettre est compressée au format JPEG de sorte qu'elle tienne dans un unique datagramme. Le but de ce codage est de simplifier au maximum les traitements à effectuer pour reconstituer l'image à afficher côté client. Les données JPEG doivent être ramenées à une taille de 64Ko maximum, ce qui est possible en jouant sur le taux de compression, et donc sur la qualité de l'image compressée. La Figure 73 donne des exemples de taux de compression obtenus sur des images caractéristiques montrant une personne à son poste de travail (Figure 72) en préservant au maximum la qualité de l'image.

Figure 72. Image typique de videoServer codée en JPEG à une résolution de 160x120

Résolution	RGB	JPEG	15 img JPEG / sec
768x576 (PAL)	1296 Ko	51 Ko	6120 Kbits/s
640x480	900 Ko	38 Ko	4560 Kbits/s
352x288 (CIF)	297 Ko	18 Ko	2160 Kbits/s
320x240 (SIF)	225 Ko	14 Ko	1680 Kbits/s
176x144 (QCIF)	74 Ko	7 Ko	840 Kbits/s
160x120 (QSIF)	56 Ko	5 Ko	600 Kbits/s
80x60	14 Ko	2 Ko	240 Kbits/s

Figure 73. Compression JPEG sur des images caractéristiques de videoServer et bande passante nécessaire pour la transmission de 15 images par seconde

La transmission des images par VSTP reprend le principe du "meilleur effort" : les images perdues ne sont pas retransmises et le client affiche directement toutes celles qu'il reçoit. Puisque la plupart des flux vidéo transmis par videoServer sont acquis en temps réel et de façon continue, la perte de quelques images passe généralement inaperçue. En cas de pertes importantes, et bien que le débit d'émission ne soit pas variable, la transmission est adaptative du point de vue de l'utilisateur : celui-ci voit les images arriver avec un débit qui dépend du taux de perte de datagrammes, mais qui lui semble constant.

VSTP ne prévoit rien pour corriger les éventuels déséquencements de datagrammes. Cependant, malgré plus de deux ans d'utilisation quotidienne, nous n'avons pas été capables de déceler visuellement de tels phénomènes. Ceci peut être lié à la taille des images que nous utilisons habituellement (CIF au maximum), au taux de rafraîchissement autorisé par notre réseau local et les réseaux académiques européens (plus de 10 images secondes) ou simplement à un manque d'attention. Il va de soit qu'un simple numéro de séquence ajouté à l'image JPEG permettrait de quantifier précisément les pertes de datagrammes. Toutefois, comme nous l'avons indiqué précédemment, nous nous intéressons

plus aux usages de la vidéo qu'aux technologies à mettre en œuvre pour la coder et la transmettre. Etant incapable de percevoir les pertes d'images, nous n'avons donc pas chercher à les mesurer.

Pendant la transmission des images par UDP, la connexion HTTP entre le serveur et le client reste ouverte et sert de canal de signalisation : lorsque le client fermer la connexion HTTP, le serveur stoppe la transmission UDP. Suivant l'exemple de RTCP, le canal de signalisation pourrait être utilisé par le client et le serveur pour échanger des rapports d'émission/réception : le client pourrait par exemple transmettre un octet toutes les n images reçues, ce qui donnerait au serveur une idée du délai de bout en bout et du taux de perte. Côté usages, la connexion HTTP pourrait servir au contrôle de la source d'images distante par le client, permettant par exemple de changer la taille des images ou le temps de pause.

2.4. VideoServer Multicast Protocol

La diffusion efficace d'un flux d'images de videoServer vers un nombre arbitraire de clients nécessite l'utilisation de protocoles réseaux de bas niveau adapté à ce type de transmission afin de ne pas dupliquer inutilement des données. Comme nous l'avons indiqué, une variante de UDP nommée *multicast* permet de diffuser un même datagramme à un groupe d'applications en un seul envoi. Nous avons donc modifié videoServer afin de pouvoir utiliser cette variante pour diffuser un flux d'images vers plusieurs clients.

La Figure 74 illustre l'utilisation du multicast UDP par videoServer :

1. Une application, appelée *initiateur*, envoie au videoServer une requête HTTP identique à celles utilisées pour VSTP mais qui spécifie un groupe multicast au lieu d'une machine unique.

2. Le videoServer renvoie à l'initiateur une réponse HTTP lui indiquant qu'il a bien reçu la requête et qu'elle n'a produit aucune donnée le concernant.

3. Le videoServer transmet par UDP multicast chaque image sous forme d'un unique datagramme contenant les données au format JPEG.

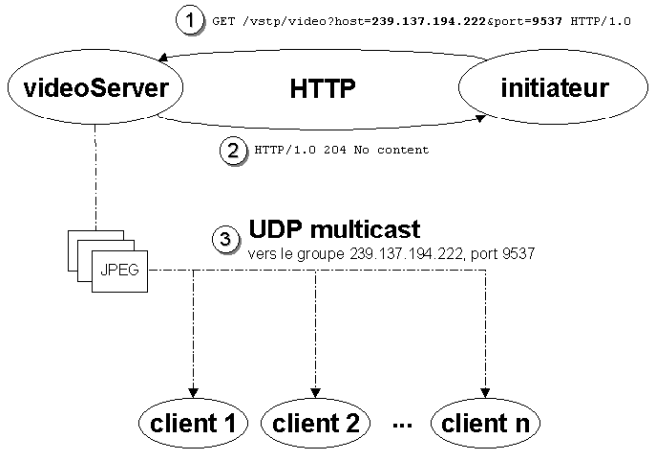


Figure 74. Principe de la diffusion d'images de videoServer par UDP multicast

Nous avons donc nommé ce nouveau protocole VSMP, pour VideoServer Multicast Protocol. Comme dans le cas de VSTP, la transmission d'images par UDP est interrompue lorsque l'initiateur ferme la connexion HTTP. L'initiateur peut bien entendu faire lui-même partie du groupe de diffusion et recevoir ainsi les images.

3. Mise en œuvre d'un système de communication basé sur videoServer

En février 1999, à l'occasion d'un séjour au Danemark, nous avons pu utiliser videoServer pour mettre en œuvre un système de communication vidéo multipoint destiné à relier deux espaces du département d'Informatique de l'Université d'Aarhus. Les deux espaces concernés étaient une cafétéria et un espace de détente situés au premier étage et au rez-de-chaussée d'un même bâtiment partagé par des chercheurs et des étudiants. La cafétéria fut équipée d'une caméra et d'un téléviseur grand écran relié à une SGI Octane placée dans un bureau adjacent, et une SGI O2 fut installée dans l'espace du rez-de-chaussée.

Une présentation fut faite dans le cadre d'un séminaire local pour expliquer le concept de mediaspace et l'installation envisagée. Afin d'éviter tout malentendu sur la présence des caméras dans les deux espaces publics, une note fut placée à côté des équipements et il fut décidé que l'accès à leurs images serait dans un premier temps limité à ces deux points seulement. Il fut également décidé que l'application utilisée pour afficher les images montrerait à la fois l'espace distant et l'espace local, favorisant ainsi la compréhension du système par les individus filmés.

L'accès restreint aux images était une solution provisoire ; nous comptions bien par la suite pouvoir étendre le système de

communication par l'équipement d'autres espaces publics ou privés et par la possibilité d'accès pour tous par le Web.

3.1. MultiVideoClient

MultiVideoClient est une application que nous avons réalisée afin de pouvoir afficher les images de plusieurs videoServers sur un moniteur analogique, comme le téléviseur de la cafétéria, ou dans une fenêtre sur un écran informatique. Cet affichage se décompose en deux zones : l'une des sources est affichée en grand, avec un label indiquant sa localisation, tandis que l'ensemble des sources disponibles est présenté sur la droite de la surface d'affichage sous forme de vues miniatures (Figure 75).

Figure 75. Exemple d'affichage de multiVideoClient. Les images montrent les prévisions météorologiques pour la région, la cafétéria et l'espace de détente.

Lorsqu'il est exécuté, multiVideoClient se connecte à un *pipe nommé*[39]. Ce *pipe* permet à d'autres applications de lui envoyer des commandes textuelles décrivant les sources à afficher ou demandant le changement de la source sélectionnée. Lorsque multiVideoClient utilise une fenêtre de l'écran informatique, il est également possible de changer la source principale en cliquant sur l'une des vues miniatures.

Chaque source est décrite par au moins une URL. Une seconde URL peut être spécifiée pour changer l'origine ou faire varier la taille ou la fréquence des images selon que la source est sélectionnée ou affichée en miniature. Une commande optionnelle peut également être spécifiée, qui sera exécutée lorsque la source est sélectionnée. Cette commande peut par exemple établir un lien audio entre l'espace local et l'espace distant au moyen d'une application comme SpeakFreely (Figure 76).

[39] Un pipe nommé est un mécanisme proposé par les systèmes UNIX pour permettre l'échange d'informations entre plusieurs applications

```
clearSources

newSource "Weather forecast"
setLowRes  http://cnn.com/WEATHER/cnn.data/europe_forecast.jpg
setHighRes http://www.meteo.fr/temps/europe/modele/carteeumod.jpg

newSource Kantine
setLowRes  vsmp://224.0.0.41:5551
setCmd     sfmike -a ozon

newSource 34C
setLowRes  vsmp://224.0.0.40:5550
setCmd     sfmike -a neapolis

selectSource Kantine
```

Figure 76. Exemple de script d'initialisation de multiVideoClient

3.2. Utilisation de videoServer et multiVideoClient

Deux videoServers et deux multiVideoClients ont été utilisés avec le protocole VSMP pour relier pendant quelques mois la cafétéria et l'espace de détente selon le schéma de la Figure 77.

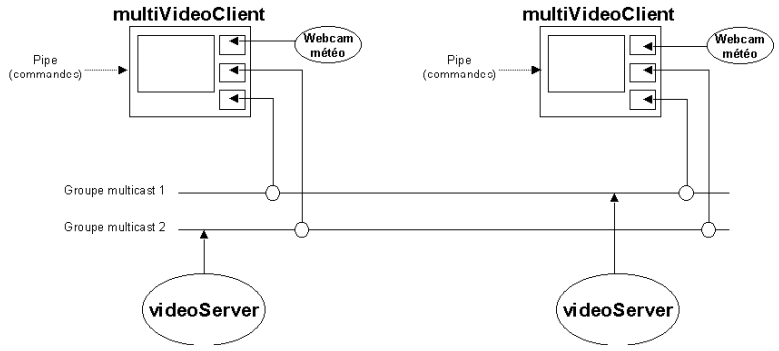


Figure 77. Liaisons utilisées entre les videoServers et les multiVideoClients pour relier la cafétéria et l'espace de détente

Le principal problème rencontré lors de l'utilisation de ce système vidéo numérique fut la difficulté de trouver un compromis acceptable entre la fluidité des liaisons vidéo et l'utilisation du réseau informatique. Les machines SGI O2 sont équipées de composants matériels destinés à la compression JPEG capable de compresser un flux vidéo PAL en temps réel (25 images 768x576 par seconde). Cette compression vidéo en temps réel génère des débits qui peuvent atteindre les 10 Mbits/sec, ce qui peut rapidement saturer un réseau local. La Figure 78 montre ainsi l'évolution de la charge moyenne du réseau local lors d'un test de transmission unidirectionnelle d'un tel flux par VSMP entre deux O2 : cette charge passe d'un niveau moyen dépassant rarement 50% au cours de la journée à plus de 80%.

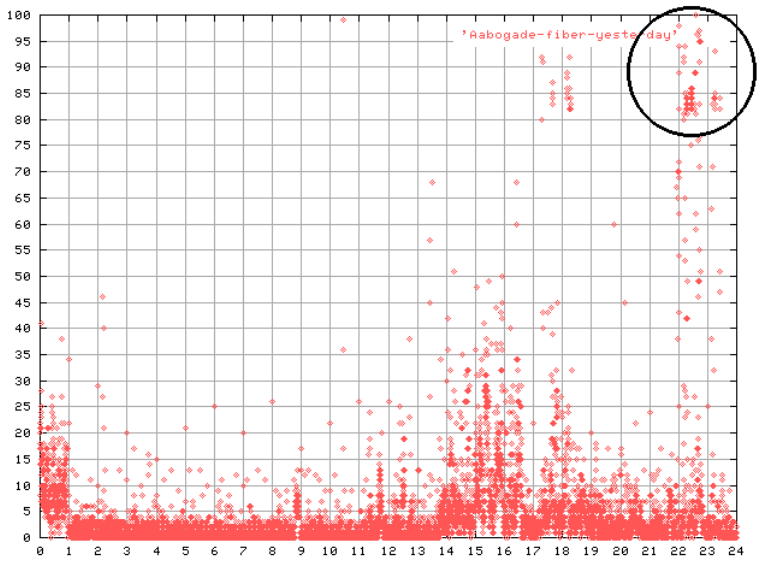


Figure 78. Transmission unidirectionnelle d'un flux vidéo PAL par VSMP entre deux O2, de 22h à minuit

Afin de limiter l'impact des liaisons vidéo permanentes sur le réseau, nous avons été contraints de réduire la taille et la fréquence des images transmises. Les flux vidéo transmis entre l'Octane de la cafétéria et l'O2 de l'espace de détente ont donc été ramenés à environ 2 images 320x240 par seconde. La Figure 79 montre l'évolution de la charge du réseau durant une transmission bidirectionnelle typique entre ces deux machines : la charge passe d'un intervalle 0-20% à 5-70%.

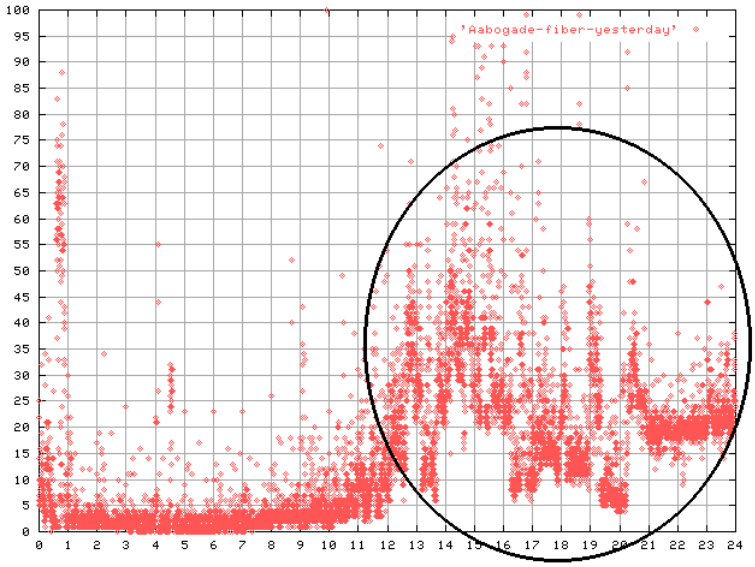


Figure 79. Transmission bidirectionnelle de 2 images 320x240 par seconde entre une O2 et une Octane

Cette expérience de mise en œuvre d'un système destiné à fonctionner en permanence souligne une fois de plus les difficultés purement techniques liées à l'utilisation de vidéo numérique. Il

serait facile de dire que l'évolution du matériel aidant, les problèmes de bande passante seront prochainement résolus. Le passage progressif de la plupart des réseaux locaux de type ethernet à des capacités de débit de 100 Mbits/sec devrait, il est vrai, faciliter ce type de transmission vidéo. Cependant, comme nous l'avons expliqué dans le Chapitre II, la communication est avant tout un phénomène social et ne peut se réduire à des mesures quantitatives.

Les liaisons entre les espaces de détente du premier Media Space de Xerox permettaient à des collègues de travail de discuter de façon informelle et quotidienne malgré la distance et d'avoir le sentiment d'appartenance à un groupe unique. Si notre expérience de liaison entre la cafétéria et l'espace de détente n'a pas été prolongée, c'est peut-être parce que la qualité de la liaison était jugée insuffisante, sans doute également par manque de support technique après notre départ, mais c'est vraisemblablement surtout parce que cette liaison n'apportait pas grand chose aux personnes concernées. La faible distance entre les deux espaces reliés, plus que les problèmes techniques, a limité l'intérêt de notre expérience. Lorsque la co-présence physique est possible, le système de communication proposé doit apporter quelque chose de plus pour être utilisé. Les chances d'apercevoir une personne sur le téléviseur de la cafétéria étaient malheureusement identiques à celles de rencontrer la même personne dans le couloir.

Quelques étudiants et chercheurs ont à plusieurs reprises exprimé l'envie de pouvoir voir les images des deux espaces depuis leur bureau, par le Web. Ce service, imaginé dès le départ et techniquement possible, n'a jamais été rendu public. Malgré les explications données sur son intérêt potentiel et les possibilités de contrôle d'accès, certains membres du laboratoire craignaient en effet l'utilisation du système par d'autres personnes et n'aimaient pas être filmés. Peut-être ce service d'accès par le Web aurait-il apporté une réelle valeur ajoutée pour les habitués du bâtiment ? La protection de la vie privée d'un groupe de personnes dans un espace public est un problème extrêmement complexe. Le principe de réciprocité stricte que nous avions choisi offrait une solution simple que les utilisateurs ont préféré conserver, malgré son incompatibilité avec l'accès depuis des postes individuels sans équipement audiovisuel.

4. Résumé du chapitre

Nous avons présenté dans ce chapitre plusieurs applications développées spécifiquement pour recevoir et afficher les images venant d'un videoServer. Nous avons également décrit deux nouveaux protocoles qui ont été ajoutés à videoServer pour permettre une transmission d'images plus efficace vers ces applications.

Nous avons montré comment videoClient permet d'étendre l'approche centrée sur les documents du Chapitre IV en faisant sortir la vidéo du navigateur Web et en facilitant son intégration à des applications nouvelles ou existantes. Nous avons également montré que d'autres applications telles que analogVideoClient peuvent être utilisées pour afficher les images de videoServer sur un moniteur vidéo au lieu de l'écran informatique, recréant ainsi les conditions d'utilisation des mediaspaces analogiques.

Nous avons décrit un exemple d'utilisation de l'un des nouveaux protocoles associé à une nouvelle application, multiVideoClient, pour relier par l'image deux espaces publics de l'Université d'Aarhus à la manière du premier Media Space. Nous avons exposé les difficultés techniques rencontrées lors de cette expérience et présenté les problèmes sociaux qui sont, selon nous, réellement responsables de son échec.

Les applications présentées dans ce chapitre constituent une base intéressante pour intégrer la vidéo dans l'environnement informatique quotidien non plus par le biais des documents, comme nous l'avons fait dans le Chapitre IV, mais par la conception de nouvelles applications ou la modification d'applications existantes. Les problèmes techniques rencontrés lors de notre séjour à l'Université d'Aarhus soulignent cependant une fois de plus la difficulté de mise en œuvre de telles applications.

Si l'on veut réellement se concentrer sur les usages de la vidéo et les problèmes sociaux qui en découlent, il faut pouvoir disposer d'un ensemble d'outils permettant d'expérimenter ces usages sans se préoccuper des problèmes techniques. Dans le chapitre suivant, nous présentons *videoSpace*, une boîte à outils logicielle pour la vidéo que nous avons développée dans cette optique.

Chapitre VI

VideoSpace

Nous avons décrit au Chapitre IV comment Mediascape et videoServer permettent aux utilisateurs d'intégrer les services de communication au sein de leurs documents HTML. Dans le Chapitre V, nous avons montré comment videoClient renforce encore cette intégration en permettant l'ajout de sous-fenêtres vidéo à des applications en cours d'exécution. Mediascape, videoServer et videoClient constituent un premier pas vers la notion d'*Ubiquitous media* [Bux97] permettant aux utilisateurs d'augmenter leur environnement habituel en y intégrant des services de communication vidéo.

Dans ce chapitre, nous nous intéressons à l'étape suivante qui consiste à offrir aux programmeurs les moyens de bénéficier eux aussi de cet espace de communication pour étendre leurs applications et développer de nouveaux usages de la vidéo.

1. Motivations

Si elle offre des perspectives intéressantes pour les applications interactives mono ou multi-utilisateur, la gestion de multiples flux vidéo en temps réel reste une tâche difficile pour le programmeur. Chaque système d'exploitation propose ses propres librairies d'accès au matériel vidéo et au système graphique, telles que Quicktime d'Apple, DirectX de Microsoft ou les Digital Media Libraries de SGI, qui sont généralement incompatibles avec celles des autres systèmes. Le nombre de flux pouvant être géré simultanément peut en outre imposer la répartition de l'application envisagée sur plusieurs machines – voir l'Annexe A –. La plupart des travaux auxquels nous avons fait référence dans les Chapitres I et II ont nécessité d'importants développements logiciels

spécifiques qui sont difficilement adaptables à d'autres contextes. Bien que nous soyons conscients de l'importance des techniques d'acquisition d'image, de transmission ou de synchronisation, ces aspects de bas niveau ne nous intéressent pas en tant que tels. Afin de pouvoir reproduire des usages de la vidéo décrits dans d'autres travaux et en expérimenter de nouveaux nous voulons donc nous appuyer sur une plate-forme logicielle permettant de faire abstraction des détails de bas niveau.

Différents environnements pour l'utilisation sur Internet de flux audio et vidéo capturés en temps réel ont été développés dans le cadre de projets de recherche européens ou américains, tels que les actions MICE, MERCI puis Meccano[40] ou le projet MASH et les travaux antérieurs de l'Université de Berkeley [MBK+97]. Contrairement aux librairies que nous avons mentionnées, ces environnements offrent des services de haut niveau comme la transmission des flux audio et vidéo, la gestion de session ou l'enregistrement et le rejeu ainsi qu'une interface de programmation. Le projet MASH, par exemple, propose un modèle de programmation à deux niveaux. La gestion de session et des flux multimédias est assurée par un ensemble d'objets élémentaires implémentés en C++. Des scripts Tcl permettent ensuite de lier et de configurer ces objets élémentaires pour décrire des applications plus complexes comme celle de la Figure 80.

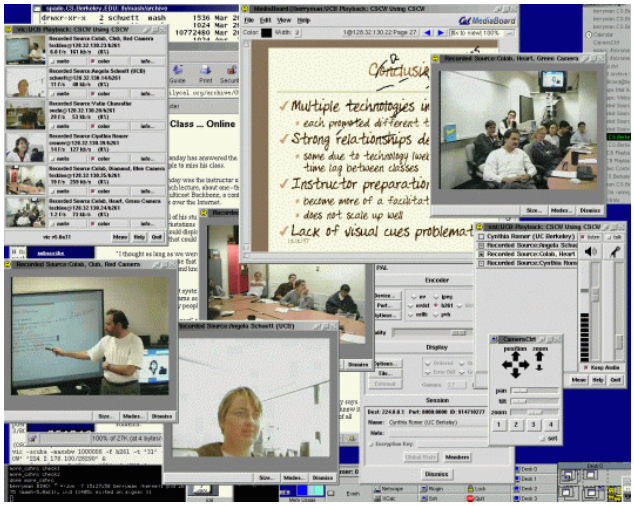


Figure 80. Exemple d'utilisation des outils du projet MASH pour du télé-enseignement (Berkeley MASH Project)

Toutefois, l'inconvénient majeur des boîtes à outils existantes pour la vidéo est qu'elles s'inscrivent toutes dans l'approche traditionnelle de la communication vidéo que nous avons décrite au Chapitre II. Elles s'attachent uniquement à des situations de communication formelle face à face de type télé-enseignement ou téléconférence et reposent donc sur un modèle simpliste de

[40] http://www-mice.cs.ucl.ac.uk/multimedia/projects/

l'utilisation des images. Ce modèle ne distingue que trois phases : acquisition, transmission et restitution à l'identique (Figure 81).

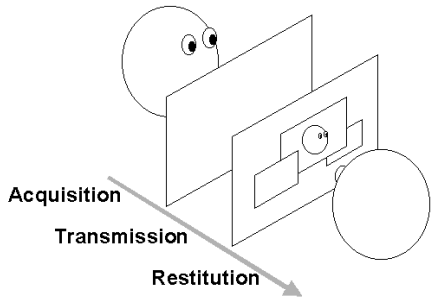


Figure 81. Modèle simpliste de l'utilisation de la vidéo dans des applications interactives

Cette vision simpliste du flux d'images au sein d'une application interactive met bien évidemment l'accent sur les aspects réseau, l'acquisition et la restitution à l'identique ne posent généralement pas de problème. La plupart des résultats de recherche liés à ces boîtes à outils portent donc sur la transmission des données, la synchronisation des flux et les mécanismes de gestion de session. La granularité des composants de base reflète également cette orientation. L'établissement d'une liaison vidéo suppose généralement de façon implicite que les images doivent être transmises le plus rapidement possible pour être ensuite affichées en l'état dans une fenêtre dédiée.

Les nouveaux usages que nous envisageons pour la vidéo reposent sur la diversification de la nature des images utilisées, la capacité de traitement en temps réel de ces images et les possibilités d'intégration de celles-ci dans des applications existantes. Le cycle de vie d'une image dans ces applications est plus complexe que le simple schéma acquisition/transmission/restitution. Les images peuvent ainsi être transmises plusieurs fois de suite, être modifiées ou ne pas être affichées. Les interfaces de programmation existantes n'offrant pas une granularité suffisamment fine pour pouvoir prendre en compte ces nouvelles utilisations des images, nous avons donc développé notre propre boîte à outils pour la vidéo : videoSpace.

2. La boîte à outils videoSpace

VideoSpace est une boite à outils logicielle destinée à faciliter l'utilisation de flux vidéo dans les applications interactives. Contrairement à la plupart des travaux similaires qui l'ont précédé, videoSpace n'a pas été conçue pour un type particulier d'application, mais au contraire pour favoriser l'émergence de nouvelles utilisations des images. En ce sens, elle est plus orientée vers la recherche en Interaction Homme-Machine ou le Collecticiel

que vers la recherche sur l'évolution des protocoles de communication sur le réseau.

VideoSpace a été initialement développée en C++ sur des machines SGI Indigo2, Indy, O2 et Octane. L'ensemble des applications et de l'API représente aujourd'hui environ 70 classes et plus de 15000 lignes de code. L'API étant conçue de façon à ce que l'on puisse aisément ajouter les classes nécessaires à la gestion de matériels vidéo spécifiques, elle peut, en théorie du moins, être porté sur d'autres systèmes de type UNIX sans trop de difficultés. En plus des systèmes SGI, videoSpace tourne aujourd'hui sur des systèmes Linux.

Comme nous l'avons vu au chapitre précédent à propos de videoClient, l'architecture du système X Window permet d'intégrer ces programmes à des applications écrites en Tcl/Tk. Certaines fonctionnalités de la librairie ont également été rendues accessibles depuis le langage Python. Malgré quelques tentatives aux résultats prometteurs, videoSpace n'a jamais réellement tourné sur les systèmes MacOs et MS Windows. Cependant, une partie de l'API a été récemment implémentée en Java, ce qui permet le développement de clients videoSpace pour cette plate-forme.

La conception de videoSpace a été guidée par l'idée que les choses simples doivent se faire simplement, mais que des choses complexes doivent également être possibles. La boîte à outils se compose d'un ensemble d'applications destinées aux utilisateurs, telles que videoServer et videoClient, et d'une interface de programmation (API). Cette API permet à des programmes C++ d'accéder à des sources d'images locales ou distantes, de traiter ces images en temps réel, puis de les stocker, les transmettre ou les afficher.

3. Interface de programmation de videoSpace

L'*image* est le concept central de l'API videoSpace. Une classe `Image` décrit des images rectangulaires caractérisées par leurs dimensions (largeur et hauteur), l'adresse d'une zone mémoire contenant les données et le codage utilisé pour représenter ces données (RGB, ABGR, RGBA, L[41] ou JPEG). Les objets de la classe Image sont utilisés comme conteneur d'échange entre les différentes opérations qui peuvent avoir lieu au cours de la vie d'une image dans l'application : production, traitement, transmission, stockage ou affichage (Figure 82).

[41] pour Luminance

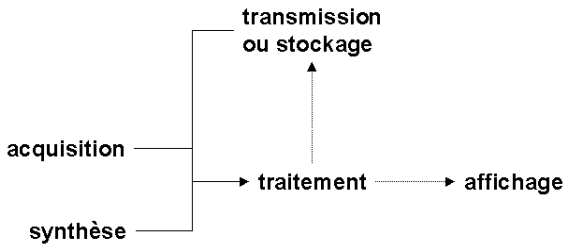


Figure 82. Cycle de vie d'une image dans videoSpace

3.1. Production d'images

Les images utilisées dans videoSpace sont produites par des sources. Ces sources peuvent être locales ou distantes et peuvent produire des images enregistrées, synthétisées ou capturées en temps réel. Chaque source d'images est décrite par une URL qui la caractérise de manière unique et permet de modifier son comportement par l'intermédiaire des paramètres optionnels.

Les fichiers contenant une image ou une séquence d'images constituent le premier type de source accessible par videoSpace. Pour le moment, ces images enregistrées doivent être codées au format JPEG, et les séquences doivent être décrites selon la syntaxe du mécanisme de server-push HTTP décrit dans le Chapitre III. D'autres formats d'image ou de séquence pourront être ajoutés par la suite, tels que PNG, GIF, Quicktime Movie ou MPEG. Ces sources enregistrées sont décrites par des URLs du type :

```
file:chemin
```

Le deuxième type de source accessible par videoSpace regroupe les périphériques vidéo connectés à la machine locale. L'URL décrivant une telle source spécifie le périphérique (carte d'acquisition, tuner TV, etc.) ainsi que l'entrée (prise caméra ou canal TV par exemple) à utiliser. Des identificateurs spéciaux, `anydev` et `anynode` permettent d'utiliser des valeurs par défaut. Diverses options peuvent être utilisées pour spécifier la taille des images, leur nombre, le temps de pause entre deux images, et leur qualité si elles sont codées au format JPEG. Ces options permettent également d'indiquer si l'accès au matériel doit être verrouillé ou s'il peut être préempté par d'autres applications. Ces sources permettant la capture d'images en temps réel sont décrites par des URLs du type :

```
videoin:/périphérique/entrée?options
```

Enfin, videoSpace permet d'accéder à des sources d'images distantes à travers différents protocoles. Quatre protocoles sont pour le moment disponibles : le server-push HTTP d'images JPEG, les protocoles VSTP et VSMP décrits au chapitre précédent, ainsi

que le protocole RFB, développé par les laboratoires AT&T de Cambridge et que nous présenterons en détail dans le chapitre suivant. Les sources distantes sont décrites par des URLs du type :

```
http://machine:port/requète?options

vstp://machine:port/requète?options

vsmp://machine:port (unicast)
vsmp://groupe:port  (multicast)

rfb://machine:port
```

Une classe nommée `ImageSource` sert de base à toutes les sources d'image. Les mécanismes d'héritage de C++ permettent, à partir de cette classe de base, de dériver les multiples classes correspondant aux combinaisons possibles de protocoles (fichier, matériel, HTTP, VSTP, VSMP, RFB, etc.) et de codage (RGB, JPEG, etc.) des images. Une fonction permet ensuite de créer une instance de la classe appropriée à partir du codage désiré et de l'URL de la source, puis de l'utiliser comme une `ImageSource` générique. Le code ci-dessous permet ainsi de créer une source d'images au format RGB à partir d'un videoServer :

```
ImageSource *src ;

src = createImageSource(Image::RGB,
                    "http://roussel.lri.fr:5555/push/video") ;
```

Une fois la source créée, différentes méthodes permettent de déclencher ou stopper la production d'images, de demander l'image suivante ou de passer directement à la plus récente dans le cas où plusieurs images auraient été stockées en mémoire par l'objet source. Le code ci-dessous montre comment obtenir une image à partir de la source que nous venons de créer :

```
src->start() ;                  // Déclenche la production
                                // d'images

Image img ;

src->waitForNextImage(&img) ;   // Bloque jusqu'à ce qu'une
                                // nouvelle image soit
                                // disponible
```

De nouveaux types de sources d'images peuvent être ajoutés à videoSpace en ajoutant d'autres classes dérivées d'`ImageSource` et en modifiant la fonction `createImageSource`. Ces nouveaux types de sources sont alors accessibles depuis toute application videoSpace grâce à l'utilisation des URLs. Il est ainsi possible d'envisager l'amélioration future des performances d'une

application qui utilise des sources en réseau par l'ajout de nouveaux protocoles mieux adaptés.

3.2. Traitement des images

VideoSpace permet le traitement des images en temps réel par l'intermédiaire de filtres. Les filtres sont des objets qui transforment ou analysent une ou plusieurs images. Ils sont définis par des classes C++, ce qui permet d'associer des données ou un état aux algorithmes nécessaires au traitement des images.

Les filtres qui transforment les images agissent sur les données ou modifient les dimensions, le codage ou l'adresse des données. La Figure 83 montre le résultat d'un tel filtre qui transforme l'image sur laquelle il est appliqué pour ne conserver que les points où elle diffère de la précédente image filtrée.

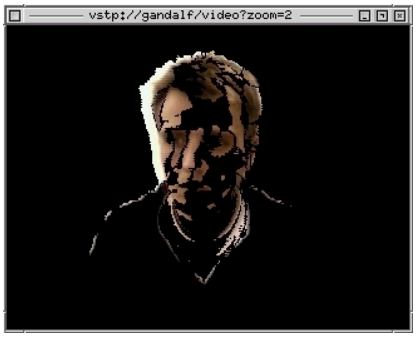


Figure 83. Exemple de filtre de transformation

Les filtres qui analysent les images ne les modifient pas mais se contentent d'en extraire des informations. La Figure 84 montre ainsi l'utilisation d'un filtre basé sur la différence entre images successives qui ne modifie pas les images mais fournit les positions et tailles des rectangles englobants les objets en mouvement. Disposant de cette description des objets en mouvement, l'application peut alors déterminer une zone d'intérêt globale et l'afficher avec plus de détails, produisant ainsi l'effet d'un zoom.

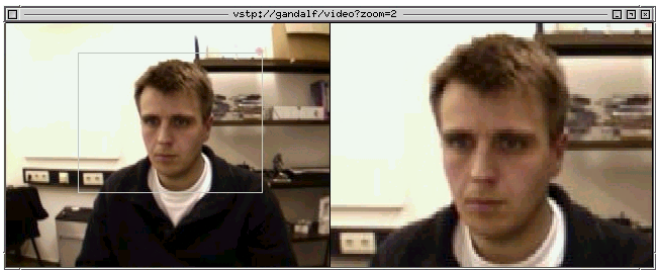


Figure 84. Exemple de filtre d'analyse

VideoSpace propose un certain nombre de filtres de transformation ou d'analyse des images. Un premier ensemble de filtres permet de passer d'un codage de données à un autre, de RGB à JPEG par

exemple, et de redimensionner les images. D'autres filtres sont utilisés pour des corrections colorimétriques (correction gamma par exemple) ou des convolutions par des filtres décrits par des matrices 3x3. Des filtres de différence d'image et de détection de mouvement sont également disponibles, ainsi qu'un filtre de chroma-keying qui permet de rendre transparents les points de l'image d'une couleur donnée[42]. Enfin quelques filtres permettent d'insérer une image dans une autre, de les superposer par transparence ou d'en colorier des régions rectangulaires.

Le code ci-dessous montre l'utilisation d'un filtre de convolution pour adoucir une image suivie d'un filtre de différence comme celui de la Figure 83 :

```
// Crée la source d'images
ImageSource *src ;
src = createImageSource(Image::RGB,
                    "http://roussel.lri.fr:5555/push/video") ;

// Déclare les deux filtres
SimpleFilter *blurFilter, *diffFilter ;

// Crée le filtre pour adoucir
blurFilter = (SimpleFilter *) new Convolution_3x3(1,1,1,
                                                  1,1,1,
                                                  1,1,1,
                                                  0.0, 9.0) ;

// Crée le filtre de différence d'image
diffFilter = (SimpleFilter *) new ImageDifference ;

Image img ;

// Tant que la source est active…
for (src->start(); src->isActive(); ) {
   // Attend une nouvelle image
   src->waitForNextImage(&img) ;
   // Applique le premier filtre sur l'image
   blurFilter->apply(&img) ;
   // Applique le second filtre sur l'image
   diffFilter->apply(&img) ;

   …
}
```

Les filtres de videoSpace sont généralement implémentés pour les codages RGB et ABGR. Les images codées selon les autres formats doivent donc être converties avant de pouvoir être traitées. Afin de faciliter la gestion dynamique de la mémoire au cours des traitements successifs, le pointeur de zone de données d'une image ne peut être modifié qu'en précisant explicitement ce qui doit advenir de la nouvelle zone lors de sa prochaine modification.

[42] Un exemple d'utilisation de ce filtre est donné au chapitre suivant

Cette information stockée avec le nouveau pointeur permet à une même image d'utiliser successivement des zones mémoire différentes, chaque zone étant au besoin libérée lorsque l'image n'en a plus besoin (Figure 85).

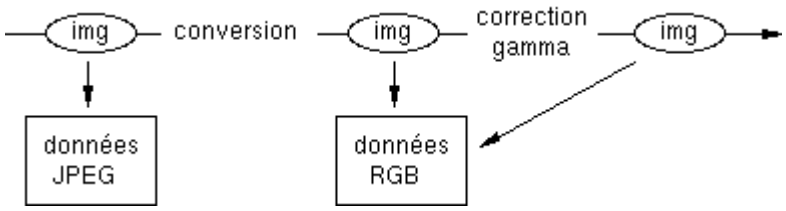


Figure 85. Exemples de transformations successives d'une image. La première transformation change l'adresse des données, la seconde ne change que le contenu

3.3. Transmission, stockage et affichage des images

VideoSpace propose plusieurs classes destinées à la communication entre applications. Ces classes permettent la transmission de datagrammes UDP, unicast ou multicast, la création de serveurs et de clients TCP ainsi que le décodage des messages HTTP. Ces classes réseau peuvent être utilisées pour transmettre tous types de données entre les applications et sont à la base des services de transmission vidéo de videoServer. Comme nous l'avons mentionné précédemment, videoSpace utilise le codage JPEG comme codage de base pour l'échange et le stockage d'images. Les flux d'images JPEG peuvent être transmis par server-push HTTP ou par VSTP et VSMP, les deux protocoles spécifiques de videoSpace. Les flux formatés pour le server-push HTTP peuvent également être sauvés dans des fichiers pour une utilisation ultérieure.

Les images de videoSpace pouvant être codées dans des formats standards comme RGB ou ABGR, elles peuvent être affichées sans aucun problème par la plupart des librairies graphiques. Il est ainsi possible d'utiliser SVGAlib[43], Xlib[44], GTK[45], OpenGL[46] et Open Inventor[47] pour afficher les images sur l'écran informatique. Sur les machines disposant d'une sortie vidéo analogique, il est également possible d'afficher les images sur un moniteur vidéo indépendant du système informatique ou de les envoyer vers la matrice de commutation d'un mediaspace analogique. En plus des traitements effectués par les filtres de videoSpace, l'utilisation de ces librairies

[43] http://www.svgalib.org/

[44] http://www.x.org/

[45] http://www.gtk.org/

[46] http://www.opengl.org/

[47] http://www.sgi.com/Technology/Inventor/

graphiques ou vidéo pour l'affichage peut introduire de nouvelles transformations intéressantes. OpenGL et Open Inventor permettent par exemple d'effectuer diverses transformations géométriques sur les images, d'utiliser la transparence pour les superposer sur d'autres objets graphiques ou de les utiliser comme texture (Figure 86).

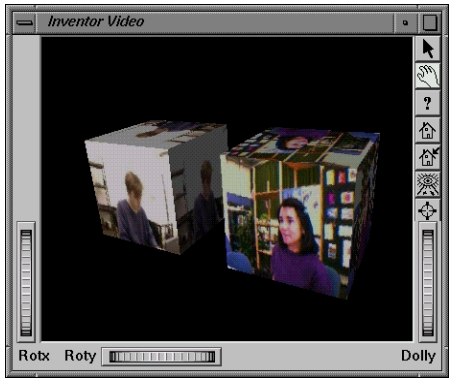


Figure 86. Affichage de flux videoSpace dans Inventor

VideoSpace propose une classe abstraite `ImageSink` destinée à faciliter la création d'applications qui se contentent d'afficher ou de stocker les flux d'images. VideoSpace contient également plusieurs classes dérivées de cette classe abstraite pour l'affichage de flux vidéo sur l'écran informatique ou sur un moniteur analogique, l'enregistrement du flux dans un fichier au format server-push ou sa diffusion par VSMP. Comme pour la désignation des sources d'images, différents schémas d'URLs sont utilisés pour pouvoir nommer ces sous-classes. Ces URLs permettent de décrire de façon simple et concise une fenêtre X Window avec affichage OpenGL ou GTK, un mode plein écran, l'utilisation de la sortie vidéo analogique, l'enregistrement dans un fichier ou la transmission par VSMP :

```
glxwindow://mimix.gmd.de:0?db=1&parent=0x1a0000e

gtkwindow://mimix.gmd.de:0?title=videoClient

svga:/640x480?centered=1

videoout:/anydev/anynode

file:/tmp/capture.vss

vsmp://mimix.gmd.de:9823    (unicast)
vsmp://225.0.0.252:5557     (multicast vers plusieurs machines)
```

Comme pour les sources d'images, l'utilisation d'URLs permet de créer les visualisateurs et enregistreurs de manière générique, quelle que soit la classe concrète utilisée :

```
Image img ;

// Crée un visualisateur (une fenêtre) GTK
ImageSink *sink = createImageSink("gtkwindow://localhost:0") ;

...

// Demande au visualisateur d'afficher l'image
sink->handle(&img) ;
```

3.4. Multiplexage

L'ajout de sources et de visualisateurs d'images peut compliquer considérablement le schéma d'exécution d'une application. L'attente de données transmises par le réseau ou le besoin de synchronisation avec le matériel vidéo posent en effet de nombreuses contraintes sur le contrôle de l'exécution. Dans le cas simple ou une source réseau est envoyée sur la sortie analogique, une image ne peut être affichée qu'une fois qu'elle a été reçue en totalité et que le matériel vidéo est prêt à la recevoir. Lorsque l'on veut combiner plusieurs sources d'images, la synchronisation entre les différents éléments devient plus complexe. Si l'on ajoute à cela une interface graphique que l'on veut interactive, la gestion des événements de cette interface vient encore compliquer le contrôle de l'exécution.

VideoSpace propose plusieurs classes destinées à faciliter la synchronisation entre les différents éléments de l'application. La première classe, `LowLevelMultiplexer`, définit des objets capables de suspendre l'exécution de l'application pendant une durée déterminée ou jusqu'au changement d'état d'un ou plusieurs descripteurs de fichier. Les descripteurs de fichiers étant l'abstraction commune des systèmes UNIX pour l'accès aux périphériques, la classe `LowLevelMultiplexer` permet donc à une application de multiplexer les accès au système de fichier, aux connections réseau et au matériel vidéo. Le code ci-dessous donne un exemple de multiplexage d'une connexion TCP qui servirait de source d'images et d'un périphérique vidéo utilisé comme visualisateur :

```
int tcp_connection, videoout ;  // Descripteurs de fichier de
                                // la connexion réseau et de la
                                // sortie vidéo
...

while (true) {

   LowLevelMultiplexer llmux ;

   // L'application doit se réveiller si des données arrivent
   llmux.setFdEvents(tcp_connection, Multiplexing::READABLE) ;

   // ou si la sortie vidéo est prête à recevoir une image
   llmux.setFdEvents(videoout, Multiplexing::WRITABLE) ;

   // Le temps maximal d'attente n'ayant pas été spécifié,
```

```
   // l'application bloque jusqu'à ce qu'une des deux
   // conditions ci-dessus soit vraie
   llmux.multiplex() ;


   if (llmux.getFdState(tcp_connection)&Multiplexing::READABLE)
   {
      // De nouvelles données sont disponibles
      … lecture de l'image …
   }

   if (llmux.getFdState(videoout)&Multiplexing::WRITABLE)
   {
      // La sortie vidéo est prête
      … affichage de l'image …
   }

}
```

Une seconde classe, `MultiplexableObject`, définit des objets réactifs capables de spécifier à un `LowLevelMultiplexer` un ensemble de descripteurs à surveiller puis de vérifier l'état de ces descripteurs après qu'un événement soit survenu. Les applications utilisant videoSpace ne manipulent généralement pas directement les descripteurs de fichiers, mais le font par l'intermédiaire d'objets `ImageSource` et `ImageSink`. Ces deux classes héritent de la classe `MultiplexableObject`. Le code ci-dessous permet de multiplexer la source et le visualisateur de l'exemple précédent sans utiliser les descripteurs de fichiers :

```
ImageSource *src ;
Image img ;
ImageSink *dst ;

bool newImage = false ;

...

while (true) {
   LowLevelMultiplexer llmux ;

   src->pre_plex(llmux) ;   // La source et le visualsateur
   dst->pre_plex(llmux) ;   // effectuent les opérations
                            // setFdEvents nécessaires avant le
                            // multiplexage

   llmux.multiplex() ;

   src->post_plex(llmux) ; // La source et le visualsateur
   dst->post_plex(llmux) ; // effectuent les opérations
                           // getFdState nécessaires après
                           // le multiplexage et changent
                           // éventuellement leur état interne

   // Demande à la source si elle a une nouvelle image
   newImage = src->getNextImage(&img) ;

   if (newImage)
```

```
        // S'il y a une image à afficher, demande au
        // visualisateur s'il peut l'afficher
        newImage = (! dst->handle(&img)) ;
    }
```

Une troisième classe, `Multiplexer`, permet d'augmenter la lisibilité de ce code en gérant une liste de `MultiplexableObject` et en exécutant automatiquement les appels `pre_plex` et `post_plex` correspondants au moment du multiplexage. Le code ci-dessous reprend l'exemple précédent en y ajoutant une seconde source, en bornant le temps d'attente et en ajoutant une connexion à un serveur X Window qui serait utilisée pour le contrôle interactif de l'application :

```
ImageSource *src1, *src2 ;
Image img1, img2, img ;
ImageSink *dst ;

// Crée les sources d'images
src1 = createImageSource(Image::RGB,
                         "vstp://gandalf.gmd.de/video") ;
src2 = createImageSource(Image::RGB,
                         "vstp://roussel.lri.fr/video") ;


// Crée un visualisateur
dst = createImageSink("videoout:/anydev/anynode") ;

// Ouvre une connexion vers le serveur X Window de la machine
Display* xDisplay = XOpenDisplay(0) ;
// Récupère le descripteur de fichier de la connexion
int xFd = ConnectionNumber(xDisplay) ;

...

// Le multiplexeur gère les deux sources et le visualisateur
Multiplexer mux ;
mux.addObject(*src1) ;
mux.addObject(*src2) ;
mux.addObject(*dst) ;

bool newImage = false ;

// Répète à l'infini…
for (;;){

   // Bloque jusqu'à ce qu'il se passe quelque chose concernant
   // le visualisateur, l'une des sources ou la connexion au
   // serveur X. L'attente est bornée à trois secondes.
   mux.reset() ;
   mux.setFdEvents(xFd, Multiplexing::READABLE) ;
   mux.setTimeOut(3000) ; // Trois seconde


   // Si plus de trois secondes se sont écoulées,
   // stoppe l'application
   int timed_out = (mux.multiplex()==0) ;
   if (timed_out) break ;
```

```
    // Traite les événements X Window en attente
    while (XPending(xDisplay)) {
       ...
    }

    // S'il y a une nouvelle image
    if (src1->getNextImage(&img1) || src2->getNextImage(&img2))
    {
       // Superpose les deux dernières images reçues pour
       // créer une nouvelle image composite
       blendImages(&img1, &img2, &img) ;
       newImage = true ;
    }

    // S'il y a une nouvelle image coimposite, demande
    // au visualisateur de l'afficher
    if (newImage) newImage = (! dst->handle(&img)) ;
}
```

Les annexes B (`miniserver.cxx`) et C (`videoClient.cxx`) donnent deux exemples concrets d'applications conçues autour d'un squelette similaire à celui-ci et capables de filtrer, d'afficher et de transmettre des images sur le réseau[48].

4. Applications de base proposées par videoSpace

VideoSpace offre aux utilisateurs un certain nombre d'applications destinées à manipuler des flux vidéo, parmi lesquelles :

- `videoServer`, que nous avons longuement décrit dans le Chapitre IV ;
- `videoClient`, que nous avons décrit dans le Chapitre V, qui a quelque peu évolué pour devenir un véritable "couteau suisse" vidéo, capable d'afficher, traiter, enregistrer ou sauvegarder un flux d'images provenant d'une source videoSpace (voir l'Annexe C).
- `multiVideoClient`, que nous avons également décrit dans le Chapitre V, qui permet d'afficher plusieurs sources d'images simultanément ;
- `grab`, qui permet de sauvegarder dans un fichier au format JPEG une image provenant d'une source quelconque ;
- `mosaic`, qui permet de composer les images de plusieurs sources pour former une mosaïque d'images, ce nouveau flux pouvant être affiché, enregistré dans un fichier ou transmis sur le réseau ;

[48] `Miniserver` est un serveur HTTP capable d'envoyer par server-push les images d'une source à ses clients.

- `videoResize`, qui permet de redimensionner les images d'un flux vidéo puis de les afficher, sauvegarder ou transmettre sur le réseau ;

- `sticky`, un script Shell UNIX qui permet de "coller" un flux vidéo affiché par un videoClient sur une application en cours d'exécution en utilisant pour cela la technique présentée dans le Chapitre V.

Quelques autres applications illustrent les différents filtres accessibles par l'interface de programmation de videoSpace et ont été utilisées pour obtenir les captures d'écran présentées dans ce chapitre.

5. Discussion

L'API et les applications qui composent aujourd'hui videoSpace sont le résultat de nombreuses itérations d'un processus alternant des phases de conception et des phases d'utilisation. Avec le recul, nous pouvons faire un certain nombre de commentaires sur les facteurs qui ont contribué à l'évolution et au succès de la boîte à outils.

5.1. La diversité des sources au service du développement incrémental

Les différents types de source d'images de videoSpace offrent une large palette de niveaux de performance en terme de fréquence et de taille des images. De ce point de vue, les sources les plus performantes sont les séquences préenregistrées et le matériel d'acquisition vidéo, capable sur certaines machines de numériser des flux au format PAL en temps réel (25 images 768x576 par seconde). Sans atteindre ce niveau, les sources réseau de videoSpace sont relativement performantes. Ainsi, lors des nombreuses transmissions vidéo que nous avons effectué entre Eindhoven (Pays Bas), Aarhus (Danemark), Bonn (Allemagne), Orsay et Dijon (France), nous avons régulièrement obtenu des performances de l'ordre de 15 à 20 images QCIF (176x144) ou 10 images CIF (320x240) par seconde avec des temps de latence inférieurs à la demi-seconde.

Ces différents niveaux de performance favorisent le prototypage et le développement incrémental des nouvelles applications. Le premier prototype d'une application peut par exemple utiliser des séquences préenregistrées et être développé sur une machine peu puissante ne disposant pas de matériel vidéo, comme un ordinateur portable par exemple. Une fois la faisabilité technique du concept validée par cette première étape, le prototype peut être testé dans des conditions plus réalistes en utilisant un flux vidéo capturé en temps réel. L'application peut ensuite être testée sur le réseau en

utilisant un videoServer distant comme source d'images. Elle peut dans ce cas bénéficier des mécanismes de contrôle et de notification de videoServer pour s'intégrer simplement aux autres applications vidéo utilisées et à la politique de contrôle d'accès choisie par l'utilisateur. Enfin, si les performances des protocoles de transmission de videoSpace s'avèrent insuffisantes, de nouveaux types de sources peuvent être ajoutés, qui seront également profitables aux autres applications.

5.2. L'importance de la hiérarchie de classes

L'un des choix de conception déterminants a été la séparation entre la classe Image, utilisée comme simple descripteur de données, et les sources, filtres, visualisateurs, transmetteurs ou enregistreurs qui manipulent ces données. L'implémentation de ces manipulateurs dans des classes distinctes de la classe Image et non comme des méthodes de celle-ci participe de façon importante à la souplesse d'utilisation et d'extension de la boîte à outils. Chaque application peut ainsi aisément définir ses propres manipulateurs qui seront créés et éventuellement modifiés au cours de l'exécution. S'ils sont par la suite réutilisés dans une autre application, ces manipulateurs peuvent alors être intégrés à la librairie commune.

L'existence de classes abstraites pour les sources, les visualisateurs et les enregistreurs/diffuseurs d'images ainsi que leur description par des URLs simplifient grandement l'écriture et l'utilisation des applications. En spécifiant les URLs adéquates, le même exécutable, et donc le même code[49], peut par exemple servir à enregistrer les images d'une caméra dans un fichier, diffuser la séquence enregistrée après filtrage ou afficher le flux diffusé :

```
Enregistre les images de la camera locale

   videoClient -i videoin:/anydev/anynode \
               -o file:demo

Diffuse la séquence après filtrage

   videoClient -i file:demo \
               -f difference \
               -o vsmp://localhost:5557

Affiche les images diffusées

   videoClient -i vsmp://localhost:5557
```

5.3. L'évolution par bootstrapping

Pourquoi l'exécutable à tout faire que nous venons de voir s'appelle-t-il encore videoClient ? Les raisons sont historiques et la

[49] Le code en question est donné en Annexe C

différence entre la fonction suggérée par ce nom et les multiples utilisations actuelles du programme illustre bien l'évolution progressive de la boîte à outils.

VideoSpace a été conçue au départ comme une librairie permettant d'accéder aux images d'une caméra capturées par videoServer ou enregistrées par videoRecorder. Une fois cette première étape atteinte, videoServer et videoRecorder ont pu être réécrit pour transmettre et enregistrer des sources distantes en plus de la caméra locale.

L'ajout de nouvelles sources par l'ajout de protocoles comme VSMP ou RFB ainsi que la création des classes abstraites de filtres, visualisateurs et enregistreurs/diffuseurs a fini par atténuer les différences entre les applications destinées à produire des images, à les traiter, les enregistrer ou les afficher. VideoServer et videoClient ont alors pu être réécrits une nouvelle fois, leur code diminuant proportionnellement à l'élévation du niveau d'abstraction auquel ils se plaçaient.

VideoServer est aujourd'hui plus un proxy HTTP qu'un serveur. Nous l'avons vu, videoClient peut remplir tous les rôles. Toute application videoSpace utilisant une `ImageSource` et un `ImageSink` possède automatiquement cette qualité.

6. Résumé du chapitre

Nous avons décrit dans ce chapitre videoSpace, notre boîte à outils logicielle pour l'intégration de la vidéo dans les systèmes informatiques interactifs. Après avoir présenté les motivations qui nous ont conduit à poursuivre les développements logiciels commencés avec videoServer et les différents videoClients, nous avons détaillé l'interface de programmation de notre boîte à outils et donné quelques exemples d'outils de base offerts aux utilisateurs. Nous avons terminé ce chapitre par une discussion sur le développement incrémental de videoSpace et les choix importants qui ont permis son évolution.

Dans le chapitre suivant, nous décrivons trois ensembles de prototypes réalisés avec videoSpace dans des contextes différents qui illustrent les possibilités actuelles offertes par cette boîte à outils et qui préfigurent les nouveaux usages de la vidéo qu'elle devrait permettre à l'avenir.

Chapitre VII

Prototypage de nouveaux usages de la vidéo avec videoSpace

La motivation principale à l'origine de la conception et du développement de videoSpace est la volonté de disposer d'un ensemble d'outils logiciels permettant de reproduire aisément des travaux de recherche antérieurs et d'expérimenter de nouveaux usages de la vidéo sans se préoccuper des détails techniques d'acquisition, de transmission ou de codage des images.

Dans ce chapitre, nous décrivons trois ensembles de prototypes réalisés avec videoSpace qui correspondent à trois usages originaux de la vidéo dans des contextes très différents. La première section du chapitre porte sur l'utilisation de l'image de la main comme télépointeur dans les applications collaboratives, la seconde décrit un système conçu pour la *téléconvivialité* et la troisième s'intéresse aux utilisations potentielles d'images provenant de l'environnement informatique.

1. La main comme télépointeur

Le geste est un moyen naturel de coordination et de communication entre individus. Les mains, en particulier, sont souvent utilisées pour exprimer des idées, se référer à des objets, attirer l'attention ou faire circuler la parole. Pour rendre possibles ce type d'échanges à distance, les collecticiels synchrones proposent généralement des télépointeurs que les participants peuvent déplacer au-dessus des objets partagés.

Les télépointeurs traditionnels sont de simples curseurs de souris. Ces curseurs sont sémantiquement pauvres : ils ont une taille et une

orientation fixe, et leur forme se restreint à un choix dans un ensemble prédéfini (Figure 87). Ces restrictions limitent la communication par gestes entre les participants. Il est par exemple difficile d'attirer l'attention ou de désigner plusieurs objets en même temps. La plupart du temps, les participants sont donc obligés d'ajouter des annotations temporaires aux objets partagés, cercles et flèches par exemple, afin de contourner ces difficultés [HPG94].

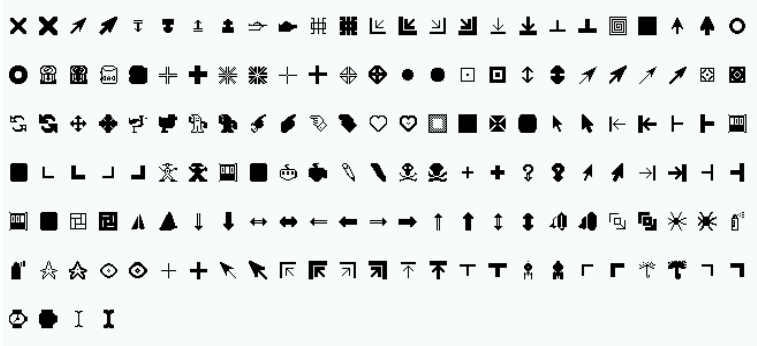


Figure 87. Les curseurs standards du système X Window

Un curseur de souris est un pauvre substitut pour le geste manuel. Les collecticiels synchrones bénéficieraient grandement de télépointeurs offrant une plus grande richesse sémantique. Cette idée n'est pas nouvelle : certains outils collaboratifs modifient déjà la taille ou la forme des télépointeurs et les complètent éventuellement par l'ajout d'informations textuelles en fonction de l'activité des participants [GGR96]. Cependant, malgré toutes ces améliorations, des travaux tels que VideoDraw ou TeamWorkStation décrits dans le Chapitre I nous conduisent à penser que la propre main de l'utilisateur serait le télépointeur idéal, à la fois intuitif et naturel.

1.1. Détourage de l'image de la main par chroma-keying

Le *chroma-keying* est un procédé utilisé en télévision et au cinéma pour insérer un fond tel qu'une carte météorologique ou un paysage derrière un sujet principal, le présentateur ou les acteurs. Pour obtenir ce résultat, le sujet est filmé sur un fond de couleur unie, généralement bleu, et les images ainsi obtenues sont traitées pour remplacer les zones de cette couleur clef par les zones correspondantes du fond à insérer. En installant une caméra au-dessus d'un bureau et en plaçant sur ce bureau un tissu ou une large feuille de papier coloré, le procédé de chroma-keying permet de détourer l'image des mains d'un utilisateur ou de tout objet posé sur le fond uni (Figure 88).

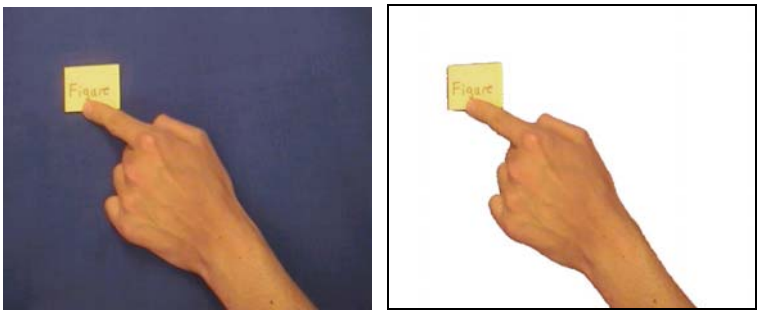


Figure 88. Avant et après le procédé de chroma-keying

VideoSpace comporte un filtre de chroma-keying logiciel. Initialement développé sur une station SGI Octane, ce filtre est capable de traiter environ 15 images CIF (352x288 pixels) par seconde selon l'algorithme suivant [Nou98, RN99]. La couleur clef est calculée comme la moyenne des couleurs présentes dans une image de référence montrant le fond uni seul. Puis, pour chaque pixel des images suivantes, on calcule sa luminosité ainsi qu'une distance entre ses composantes RGB et celles de la couleur clef. Ces deux valeurs permettent de calculer le niveau d'opacité du pixel :

- si le pixel est sombre, il peut correspondre à une ombre ou à un objet foncé. L'opacité sera proportionnelle à la distance colorimétrique ;

- si le pixel est clair et suffisamment proche de la couleur clef, le pixel est considéré comme faisant partie du fond et son opacité sera nulle ;

- sinon, l'opacité est fixée au maximum autorisé par l'utilisateur.

Ces trois cas permettent de différencier le fond à ignorer, le premier plan et, dans une certaine mesure, les ombres, certains objets foncés posés sur le fond pouvant cependant être mal interprétés. Afin de pouvoir calibrer l'algorithme, deux seuils de tolérance sont utilisés pour déterminer si un pixel est sombre, et s'il est suffisamment proche de la couleur clef.

Le chroma-keying appliqué à des images comme celles de la Figure 88 offrirait sans doute plus de possibilités de communication entre des utilisateurs distants que de simples télépointeurs. Il permettrait en effet d'utiliser les mains pour communiquer par gestes et d'annoter la vue partagée à l'aide d'objets réels. Afin de pouvoir tester de manière informelle l'utilisabilité des images vidéo détourées et leur intégration à des applications nouvelles ou existantes, nous avons réalisé successivement trois prototypes mettant en œuvre le procédé de chroma-keying que nous venons de décrire.

1.2. Premier prototype : simulation sur un fond d'images vidéo

Notre premier prototype utilise la librairie graphique OpenGL pour superposer deux flux d'images videoSpace : le premier flux montre ce qui est supposé être la vue de l'application partagée, tandis que le second montre la vue d'une caméra placée au-dessus du bureau de l'utilisateur. Les images provenant de cette caméra sont traitées par l'algorithme de chroma-keying puis superposées sur le fond grâce à la gestion de l'opacité/transparence d'OpenGL. Ce prototype peut être utilisé avec un fond fixe (Figure 89) ou montrant une séquence préenregistrée d'utilisation d'une application réelle.

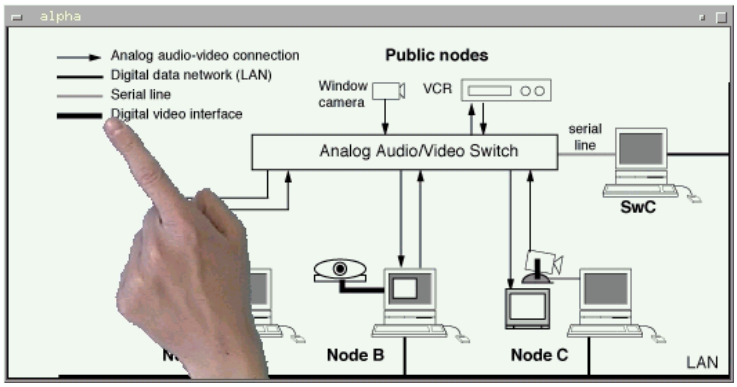


Figure 89. Superposition d'un flux vidéo détouré sur un fond fixe

Lorsque l'image de la main apparaît à l'écran, elle est souvent trop grande, ce qui occulte une trop grande partie de l'image de fond et rend la désignation de petits objets difficile. Pour pallier ce problème, le prototype permet à l'utilisateur de régler le niveau de transparence et la taille des images superposées (Figure 90). Si elle permet une désignation plus précise, la modification de la taille des images superposées pose un nouveau problème : certaines parties de la vue partagée peuvent se retrouver hors du champ couvert par la caméra et être ainsi inaccessibles. Le prototype permet donc d'utiliser un dispositif de pointage, une souris ou un trackball par exemple, pour déplacer le flux vidéo détouré sur le fond. Ce dispositif peut également servir au contrôle de la taille des images, en utilisant par exemple deux boutons pour le zoom avant/arrière, tandis que le clavier permet d'ajuster les valeurs des seuils utilisés par le chroma-keying et le niveau de transparence du flux détouré. Ces ajustements sont en effet souvent nécessaires pour pouvoir adapter le prototype aux conditions d'éclairage particulières.

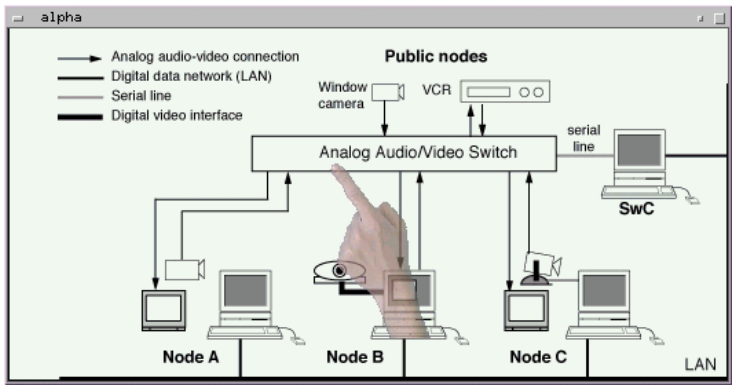


Figure 90. Après un changement de taille, de position et du niveau transparence

1.3. Discussion sur le premier prototype

Notre premier prototype permet l'utilisation d'images vidéo détourées comme pointeur sur un fond fixe ou préenregistré. Comme nous l'avons vu, la taille et la position du pointeur sont contrôlées par un dispositif de pointage, et sa transparence peut être modifiée par le clavier. Plusieurs tests informels réalisés avec différents utilisateurs semblent montrer l'efficacité de ce système et sa rapide prise en main par les utilisateurs.

Le télépointeur traditionnel remplit de nombreuses fonctions dans les collecticiels. En plus du support à la communication gestuelle, il renseigne sur l'activité du groupe en montrant la position des différents utilisateurs et permet de les identifier, généralement par l'utilisation de couleurs. Peut-on reconnaître un utilisateur à ses mains ? Que va-t-on voir s'il tape au clavier ? Sans doute faut-il combiner la technique du chroma-keying avec d'autres existantes ou à inventer. Les premiers tests réalisés nous permettent toutefois de faire quelques commentaires sur son utilisation et sa mise en œuvre.

Contrôle de la transparence, de la taille et de la position du pointeur

La transparence du pointeur vidéo permet à l'utilisateur du prototype de voir à la fois ses mains et les objets réels ainsi que les objets graphiques du fond. En ajustant le niveau de transparence, l'utilisateur arrive généralement à trouver un compromis entre la lisibilité du fond et celle du pointeur vidéo. Le traitement par chroma-keying permet de ne conserver que les éléments pertinents des images superposées, à savoir les mains de l'utilisateur et les objets utilisés pour annoter le fond. L'image composite affichée par le prototype ne présente donc pas les problèmes de perte de contraste et de luminosité caractéristiques des systèmes comme ClearBoard, TeamWorkstation ou VideoDraw qui fusionnent les images dans leur totalité. [VLNC99] propose par ailleurs une méthode pour remédier à ces problèmes de contraste et de

luminosité liés à la superposition d'images. Cette méthode repose sur l'ajustement de l'histogramme de couleur de l'image composite et pourrait éventuellement être utilisée pour améliorer la lisibilité du fond situé derrière le pointeur vidéo détouré.

En plus des formes habituelles de communication gestuelle à une ou deux mains, la possibilité d'agrandir ou de diminuer la taille du pointeur vidéo tout en le déplaçant offre de nouvelles possibilités. Il est ainsi possible de désigner un groupe d'objets en les "prenant dans sa main", en ajustant la taille et la position de l'image de la main pour qu'elle couvre tous les objets. Ce contrôle est également utile pour l'annotation par l'ajout d'objets réels, des PostIts par exemple, que l'on peut disposer précisément.

Interaction bi-manuelle pour la communication gestuelle

L'utilisation d'un dispositif de pointage manuel pour contrôler la position et la taille du pointeur vidéo transforme les tâches de télépointage classiques en tâches bi-manuelles, une main utilisant le dispositif tandis que l'autre sert à désigner. Les travaux de Y. Guiard sur le modèle de la chaîne cinématique pour la division du travail bi-manuel montrent que la main non dominante fournit habituellement le cadre général de référence dans lequel la main dominante agit à une échelle spatio-temporelle plus fine [Gui87]. Selon ce modèle, un utilisateur droitier de notre prototype devrait donc contrôler la position du pointeur avec sa main gauche et utiliser sa main droite pour désigner. Ceci implique donc l'utilisation d'un périphérique de pointage autre que celui habituellement utilisé pour interagir avec le système graphique, celui-ci étant généralement manipulé par la main dominante et se trouvant donc du mauvais côté du clavier.

Malgré les nombreux travaux tels que [BM86] ou [LZB98] démontrant la supériorité de l'interaction bi-manuelle pour de nombreuses tâches informatiques quotidiennes, l'utilisation simultanée de plusieurs périphériques de pointage reste difficile sur la majorité des plates-formes actuelles [Cha94]. Faute de support logiciel adéquat, la plupart de nos essais ont donc été réalisés en contrôlant le pointeur avec la main dominante et en désignant avec la main non dominante. Ceci explique le fait que la plupart des captures d'écran de cette section montrent une main gauche...

Traitements d'image ou matériel additionnels

On peut penser que si les utilisateurs d'un collecticiel peuvent communiquer et désigner des objets avec leurs mains, ils voudront sans doute aussi pouvoir créer, déplacer ou modifier les objets de cette façon. Ce passage du geste sémiotique au geste ergotique [Cad94] nécessitera sans aucun doute des traitements d'image ou matériels additionnels pour capturer les gestes de l'utilisateur et les interpréter.

Le *tableau magique* [Bér99] combine plusieurs traitements d'image qui lui permettent de suivre le doigt de l'utilisateur d'un tableau blanc et de déclencher par des gestes simples des commandes telles que la sélection, la copie ou la sauvegarde du contenu du tableau. Dans le cadre de notre télépointeur vidéo, la reconnaissance de gestes par ce type de techniques permettrait, outre le contrôle de l'application partagée, de contrôler la position, la taille et la transparence du flux vidéo superposé.

Nous avons expérimenté l'utilisation d'un filtre de détection de mouvement pour contrôler le positionnement du pointeur vidéo. Notre idée était de pouvoir déplacer automatiquement le pointeur lorsque la main atteint la limite du champ de la caméra. A l'utilisation, cette détection de mouvement s'est pourtant révélée moins pratique que le positionnement par dispositif de pointage, en raison du manque de souplesse causé par son caractère automatique. Cette technique a donc été abandonnée.

Notre premier prototype nous a également permis de commencer à explorer l'utilisation de tablettes graphiques. En plaçant une tablette sous le fond uni filmé, il est en effet possible d'utiliser un stylo ou tout autre élément que la tablette reconnaît comme dispositif de contrôle du pointeur vidéo. Cette technique s'est révélée particulièrement intéressante, les deux mains de l'utilisateur pouvant être dans le champ de la caméra (Figure 91).

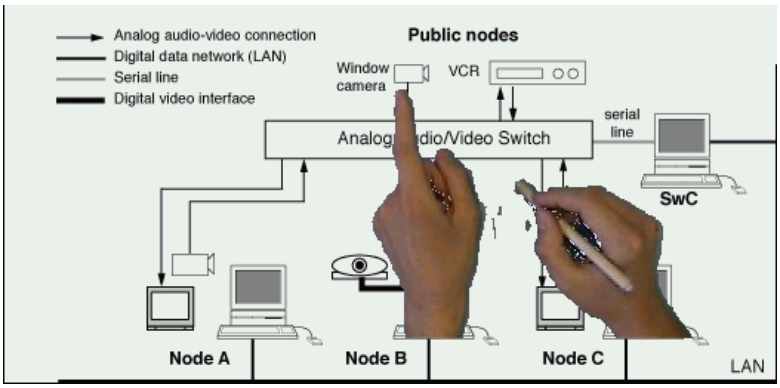


Figure 91. Contrôle du pointeur vidéo par une tablette placée sous le fond uni

Nous avons enfin imaginé une autre utilisation de la détection de mouvement qui permettrait d'isoler les objets ou annotations immobiles comme la note de la Figure 88 des mains de l'utilisateur. L'image obtenue par chroma-keying pourrait ainsi être séparée en deux plans, ce qui permettrait de limiter les changements de taille et de position aux mains de l'utilisateur. Cette dernière technique d'interaction n'a toutefois jamais été testée.

Mise en œuvre dans un contexte réel

Notre premier prototype nous a permis de tester l'utilisation d'un flux vidéo comme pointeur et de vérifier l'intérêt et la faisabilité de cette démarche. Cependant, dans ce prototype, la vue partagée est simulée par un flux vidéo et reste donc limitée à l'utilisation d'une image fixe ou de séquences vidéo préenregistrées. Nous nous sommes donc naturellement penchés sur les problèmes de mise en œuvre de notre technique dans des contextes d'applications réelles.

Si l'on a accès au code source de l'application que l'on veut modifier, les changements à effectuer sont relativement simples. VideoSpace permettant d'accéder à des sources d'images locales et distantes, il suffit donc à l'application de transmettre de son côté la position, la taille et l'opacité des pointeurs vidéo. Si l'application utilise déjà des télépointeurs traditionnels, cette intégration doit pouvoir se faire sans trop de problème, par simple extension des mécanismes de transmissions de position existants. Pour ce qui est de l'affichage des pointeurs vidéos, notre prototype montre l'exemple pour les applications utilisant OpenGL. Toute autre librairie graphique gérant la transparence, même par l'utilisation d'un masque binaire, pourrait aussi bien convenir.

La version X Window de l'éditeur Emacs est capable d'ouvrir une fenêtre sur une machine distante pour y afficher un document en cours d'édition. La personne distante peut alors voir et éditer le document qui lui est présenté. Emacs ne proposant pas de mécanisme de télépointeur, le seul moyen de communication entre les deux utilisateurs est le document lui-même, ce qui les oblige à modifier son contenu ne serait-ce que pour indiquer la partie à laquelle ils s'intéressent. Cet exemple montre bien l'intérêt qu'il y aurait à pouvoir utiliser notre technique de pointage vidéo sur des applications en cours d'exécution. Dans un contexte plus général, cette approche pourrait également être utilisée pour les applications dont le code source est inaccessible. Nous avons donc poursuivi nos travaux sur les pointeurs vidéo en concevant un nouveau prototype destiné cette fois-ci à superposer le flux vidéo détouré par chroma-keying sur une application en cours d'utilisation.

1.4. Second prototype : superposition du pointeur vidéo sur une application en cours d'exécution

Certains serveurs X Window disposent d'un plan graphique spécial appelé *overlay plane* qui permet à une application d'afficher des éléments graphiques au-dessus des fenêtres présentes à l'écran. Ce plan graphique est notamment utilisé sur certaines machines pour afficher des menus déroulants ou des rectangles de sélection sans nécessiter le réaffichage des applications temporairement masquées. L'overlay plane est généralement implémenté sur la carte graphique de la machine, leur profondeur est donc généralement limitée à 8 bits au plus, soit 256 couleurs.

Contrairement à celui d'OpenGL, le modèle graphique de X n'intègre pas la notion d'opacité variable. La semi-transparence est donc impossible. Toutefois, dans le cas de l'overlay plane, une couleur spéciale permet d'indiquer les pixels d'une image qui doivent être transparents.

Notre second prototype utilise l'overlay plane X Window disponible sur la plupart des machines SGI pour superposer des images vidéo traitées par chroma-keying sur la fenêtre d'une application en cours d'exécution. Le prototype demande à l'utilisateur de sélectionner l'application qui servira de fond, l'application hôte, en cliquant sur celle-ci. Il en détermine alors la fenêtre principale et crée une nouvelle fenêtre fille de celle-ci dans l'overlay plane. Après le traitement par chroma-keying, les images vidéo subissent une réduction du nombre de couleurs pour pouvoir être affichées quelle que soit la profondeur de l'overlay plane. La semi-transparence étant impossible, les mains, les objets et les ombres masquent l'application (Figure 92).

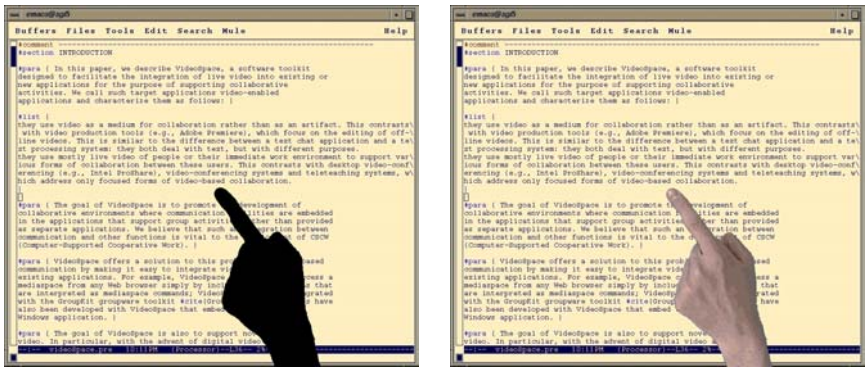


Figure 92. Second prototype utilisé avec des overlay planes de profondeur 2 et 8

Si l'affichage du pointeur vidéo ne pose pas de problème, la gestion des événements liés aux périphériques d'entrée est plus délicate. Les événements correspondant au dispositif de pointage principal sont envoyés par le serveur X à l'application qui gère la fenêtre immédiatement située sous le pointeur. La zone de l'application hôte couverte par le pointeur vidéo est donc "inactive". Ainsi, si l'utilisateur clique sur un bouton de l'application hôte qui se trouve sous le pointeur vidéo, les événements générés sont envoyés à notre prototype, et non au bouton en question. X Window permet de synthétiser des événements et de les envoyer à une fenêtre. Toutefois, la recherche de la fenêtre concernée parmi toutes celles présentes à l'écran, le bouton dans notre exemple, est incompatible avec une exécution en temps réel de l'application. Il faut donc trouver un compromis entre le contrôle du flux vidéo superposé et le contrôle de l'application hôte.

Notre prototype s'abonne aux événements correspondant à la pression des boutons et au déplacement de la souris pour permettre à l'utilisateur de positionner et redimensionner le pointeur vidéo

sur l'application. La taille de sa fenêtre est ajustée automatiquement pour être identique à celle du flux vidéo superposé, minimisant ainsi la taille de la zone "inactive" de l'application. Le prototype s'abonne également aux événements correspondant au relâchement de touche pour le réglage des seuils et de l'opacité, ces événements étant peu souvent utilisés dans les autres applications[50].

Ce second prototype illustre bien la difficulté d'intégration de techniques d'interaction innovantes dans les environnements graphiques actuels. Comme nous l'avons vu sur cet exemple, les environnements actuels reposent pour la plupart sur des modèles graphiques aujourd'hui dépassés par les capacités du matériel. La mise en œuvre de techniques d'interaction bi-manuelles ou utilisant la transparence nécessite donc l'utilisation d'extensions spécifiques, comme la *X Input Extension* ou les overlay planes, qui compliquent la réalisation des nouveaux développements et limitent leur portée. Puisqu'il est difficile d'intégrer notre pointeur vidéo aux environnements graphiques existants, nous nous sommes donc intéressés à la seule alternative restante : intégrer ces environnements graphiques à notre prototype.

1.5. Troisième prototype : insertion des applications sous le pointeur vidéo

En 1994, une équipe du laboratoire AT&T de Cambridge, alors Olivetti Research Limited, a mis au point Videotile[51], un dispositif connectable à un réseau ATM et constitué d'un écran couleur à cristaux liquides et d'un stylo. Le but initial de Videotile était de pouvoir afficher des flux vidéo sur un dispositif léger et autonome, mais très rapidement, ses concepteurs ont eu l'idée de l'utiliser également comme moyen d'interaction avec les applications exécutées sur d'autres machines. Ils ont donc mis au point un protocole, nommé *RFB* (*Remote Frame Buffer*) destiné à permettre l'échange d'images informatiques et d'événements clavier/souris entre deux machines. Contrairement à d'autres protocoles autorisant l'affichage déporté, RFB est indépendant du système d'exploitation, du système de fenêtrage et des applications utilisés. Alors que le protocole X, par exemple, permet de transmettre diverses commandes graphiques qui sont exécutées à distance pour produire une image, RFB repose sur une simple et unique primitive – l'affichage d'un rectangle de dimensions, position et couleur données – et propose plusieurs algorithmes de codage qui permettent de décrire une image existante à partir de cette unique primitive afin de la reproduire à distance. Aucune supposition n'est faite sur la nature des images, bien que les codages utilisés soient

[50] L'événement KeyPress est souvent préféré à KeyRelease car il permet de bénéficier de la répétition automatique

[51] http://www.uk.research.att.com/tile.html

plus efficaces sur des contenus graphiques uniformes. RFB est donc bien adapté à des images d'applications classiques, mais peu adapté à des flux vidéo.

RFB est à la base du système *VNC* (*Virtual Network Computing* [RSFWH98]), système qui permet à un utilisateur d'accéder à son environnement de travail informatique depuis n'importe quelle machine à travers une simple application cliente (Figure 93). Le terme visualisateur est souvent utilisé par les concepteurs de VNC au lieu du terme client. La partie cliente de VNC est en effet extrêmement simple : il suffit d'afficher les images décrites sous forme de successions de rectangles et de transmettre les événements clavier/souris. Des visualisateurs VNC sont disponibles sur les trois principales plates-formes (MS Windows, Macintosh, X Window) et une implémentation Java permet d'afficher les images d'un serveur VNC dans un navigateur Web.

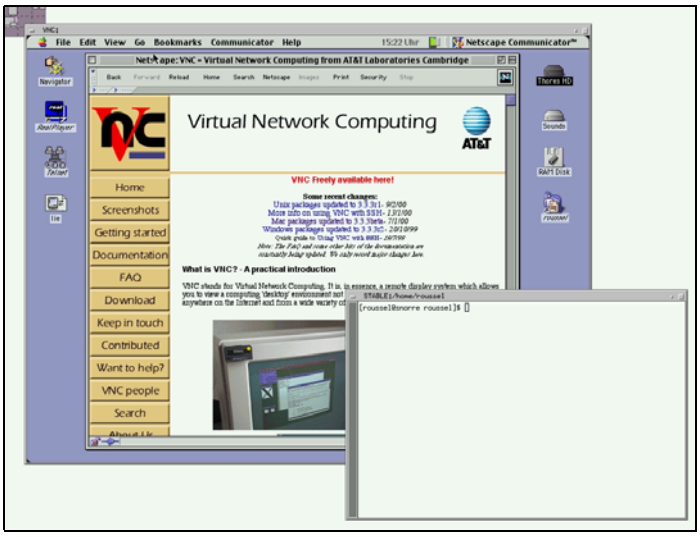


Figure 93. Environnement Macintosh accessible par RFB depuis un client VNC sous X Window

La partie la plus complexe de VNC réside dans le serveur. Celui-ci doit produire les images de l'environnement de l'utilisateur, les coder et les transmettre au client, et doit également distribuer les événements clavier/souris venant du client aux applications. Le serveur VNC disponible pour X Window, nommé Xvnc, est en fait un authentique serveur X Window qui a été modifié pour ne plus afficher les applications à l'écran mais rendre leur image accessible par RFB (Figure 94). L'environnement accessible par Xvnc peut donc être différent de celui affiché à l'écran, qui lui est géré par un serveur X classique. Les serveurs VNC pour Macintosh et MS Windows ne permettent pas cette distinction et se contentent de rendre accessible le contenu de l'écran.

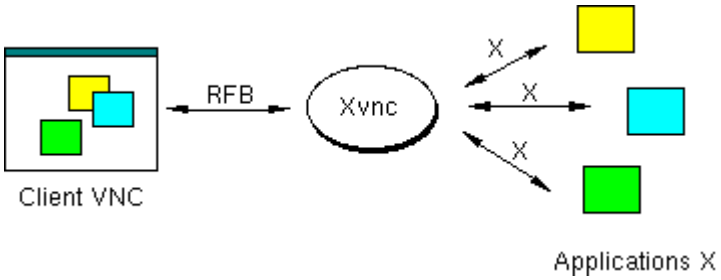


Figure 94. Principe de fonctionnement de Xvnc

En implémentant la partie cliente du protocole RFB, nous avons pu faire de tout serveur VNC une source d'image supplémentaire pour les applications de videoSpace. La classe représentant ces sources est particulière puisqu'elle permet, en plus de la réception des images, d'envoyer des événements clavier/souris au serveur distant. Ce client RFB intégré à videoSpace nous a permis de développer un troisième prototype qui affiche les images d'un serveur VNC et superpose en transparence le pointeur vidéo, en utilisant OpenGL de façon similaire au premier prototype (Figure 95). La partie clavier/souris du client RFB permet à l'utilisateur d'interagir avec les applications à travers notre prototype. La gestion des événements est beaucoup plus simple que dans le cas du second prototype : notre application reçoit tous les événements clavier/souris et peut décider si elle doit les traiter elle-même ou les envoyer au serveur VNC qui les distribuera aux applications concernées.

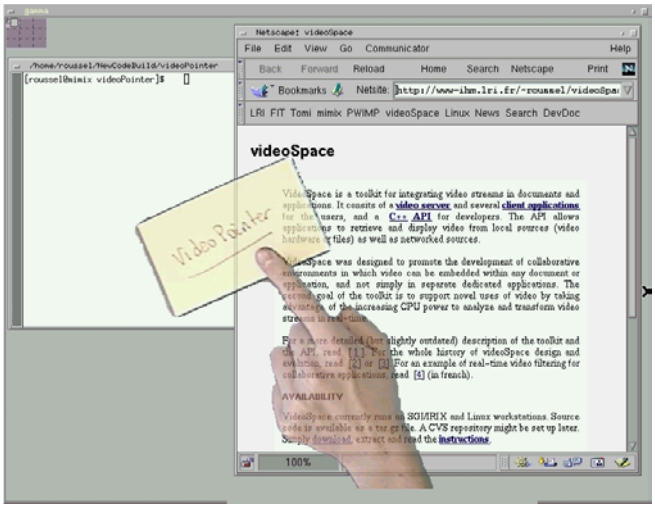


Figure 95. Troisième prototype utilisant VNC

Ce troisième prototype est très proche d'une application complète de partage de vue à distance. Pour que plusieurs utilisateurs puissent utiliser un tel système, il faudrait ajouter autant de flux vidéo détourés que de participants distants et mettre au point un système de diffusion de la position, taille et transparence des différents pointeurs. VNC et videoSpace étant conçus pour le

fonctionnement en réseau, la transmission des différentes images ne devrait poser aucun problème.

2. Vers la téléconvivialité

Les systèmes de visioconférence ou de téléréunion existent depuis de nombreuses années. La qualité de restitution de l'image et du son des systèmes les plus avancés est quasi-parfaite. L'image reproduite en taille réelle, le son spatialisé ou les transmissions multipoint deviennent peu à peu monnaie courante et rendent les expériences de communication vidéo de plus en plus agréables. L'un des regrets pourtant exprimés par les concepteurs et utilisateurs de ces systèmes de groupe concerne leur inadaptation à des formes de communication plus informelles. La disposition et la nature des équipements se prêtent en effet généralement mal aux activités se déroulant avant et après une réunion ou pendant une pause.

Figure 96. Vous voulez boire quelque chose ? (extrait de [TB94])

La position assise, face à une rangée de caméras et d'écrans n'est pas idéale pour engager une conversation informelle avec les personnes des autres sites (Figure 96). Les communications informelles pendant les pauses des conférences ou réunions se déroulent donc généralement de manière indépendante sur chaque site, n'impliquant que très peu les personnes distantes[52].

Le principal frein des systèmes de visioconférence ou de téléréunion pour la convivialité est donc, selon nous, précisément ce qui fait leur force pour les situations formelles : ils concentrent la communication entre les individus en un point unique et toujours visible et audible de tous. Nous pensons que l'un des éléments clés de la convivialité est la possibilité offerte aux individus de se déplacer et de former des petits groupes isolés les uns des autres. D'autres éléments plus subjectifs comme l'humour ou le jeu contribuent également à l'impression de convivialité. La

[52] Ceci explique sans doute l'échec de la tentative de soirée d'Halloween multi-sites évoqué par [Rie96].

téléconvivialité passe donc par l'utilisation de multiples équipements capables d'attirer les individus en plusieurs points de l'espace local où ils pourront communiquer entre eux et avec certaines personnes des sites distants.

2.1. Comment attirer les personnes vers un point particulier ?

A l'occasion de SIGGRAPH'97, T. Darrel et al. ont présenté une installation appelée *Mass Hallucinations*[53] [DGH+98]. Cette installation reposait sur un système de vision par ordinateur qui permettait de créer un miroir déformant dont les transformations se limitaient au visage des personnes filmées (Figure 97). Cette démonstration du système à SIGGRAPH a permis à ses auteurs de le tester sur environ 6000 personnes sur une période de six jours. Le véritable but de T. Darrel et de ses collègues était bien entendu la réalisation d'un système de suivi de personne par traitement d'images vidéo. Le choix de Mass Hallucinations comme application leur a permis de bénéficier de l'intérêt du public et d'obtenir ainsi les données d'utilisations nécessaires à la validation de leur système.

Figure 97. Mass Hallucinations (extrait de [DGH+98])

Les installations de M. Fleischmann sont destinées à "redécouvrir l'usage des sens par l'intermédiaire de la technologie" [FS98]. *Liquid views*[54] par exemple, s'intéresse au toucher et à l'image de soi par l'utilisation d'une caméra et d'une projection horizontale montrant à une personne son propre reflet, associées à un dispositif tactile lui permettant de perturber la surface de projection, créant ainsi l'illusion d'une surface liquide (Figure 98). Cette installation, présentée à de nombreuses reprises et plusieurs fois primée, joue une nouvelle fois sur l'attrait exercé par l'image, le dispositif étant conçu comme "une invitation à communiquer".

---

[53] http://www.interval.com/frameset.cgi?projects/hallucinations/Hallucinations.html

[54] http://viswiz.gmd.de:8080/IMF/liquid.html

Figure 98. Liquid Views (GMD, IMK.MARS)

Comme beaucoup d'autres installations présentées dans des musées ou à l'occasion d'événements exceptionnels, ces deux exemples montrent que l'image de soi ou des autres présentée de façon inattendue attire irrésistiblement le public et le pousse à vouloir interagir avec l'image. Ce type d'installation vidéo proche du jeu ou de l'œuvre d'art pourrait donc être utilisé pour attirer les participants d'une visioconférence ou téléréunion vers un dispositif plus adapté à la communication informelle. Reste une question importante : comment présenter les images des différents sites pour qu'un groupe de personnes puisse se constituer au niveau du dispositif et que la communication soit possible ?

2.2. Quelle disposition d'image pour communiquer sur plusieurs sites à la fois ?

L'*Interactable*[55] [SGH+99] de N. Streitz et al. permet à un groupe de personnes de se réunir autour d'une projection horizontale avec écran tactile pour interagir avec un système informatique (Figure 99). Ce système n'est pas destiné à la communication à distance, mais à la collaboration de plusieurs personnes sur un même site autour d'objets informatiques. L'intérêt de la projection horizontale dans ce cas est d'inciter les personnes à se répartir en cercle autour du dispositif, ce qui leur permet de mieux percevoir l'espace partagé.

Figure 99. L'InteracTable (GMD, IPSI)

[55] http://www.darmstadt.gmd.de/ambiente/activities/interactable.html

Une projection horizontale devrait faciliter le regroupement des personnes autour de notre dispositif de communication. Dans le cas d'une liaison entre deux sites, l'affichage des images ne devrait pas poser de problèmes. Mais comment disposer les images provenant de plusieurs sites ? VideoPlace a montré que lorsque des gens voient leur propre image mélangée à celle d'autres personnes, chacun se déplace naturellement de façon à éviter le recouvrement ou même le contact entre les images. Dans le cas d'une liaison entre plus de deux sites, les images des différents sites peuvent donc être fusionnées par transparence, montrant ainsi à chacun où il est et lui offrant la possibilité de se déplacer pour ne pas gêner les autres.

2.3. Le puits

Le puits est un dispositif réalisé en collaboration avec l'équipe iMagis du laboratoire GRAVIR-IMAG et le service acoustique du CSTB (Centre Scientifique et Technique du Bâtiment). La conception de ce dispositif s'inscrit dans la démarche de téléconvivialité que nous venons de développer : le puits est conçu pour attirer un petit groupe de personnes co-localisées autour d'un système de communication audio/vidéo afin de leur permettre de communiquer avec des personnes de sites distants. La conception du puits s'inspire également de certains vases que l'on peut trouver dans les bosquets du parc du château de Versailles.

Le puits se présente sous la forme d'une table haute de préférence circulaire ou ovale (Figure 100). Sa partie vidéo se compose d'un ensemble de caméras et d'un système de projection horizontal qui affiche une image composée à partir des caméras des différents puits connectés. Sa partie audio se compose d'un ensemble de microphones et de haut-parleurs répartis sur son périmètre qui permettent de créer une ambiance sonore spatialisée cohérente avec l'image affichée. La hauteur du puits invite les participants à se pencher légèrement vers l'avant pour voir l'image composée et leur permet éventuellement de s'accouder pour discuter avec leurs voisins immédiats. Afin de limiter les problèmes de cadrage vidéo et de prise de son, les personnes situées autour du puits doivent être guidées "naturellement" vers les positions correspondant aux conditions idéales. Les micros, haut-parleurs et caméras sont donc placés de façon à ce que les participants soient dans le champ des caméras pour voir et proches des micros pour entendre.

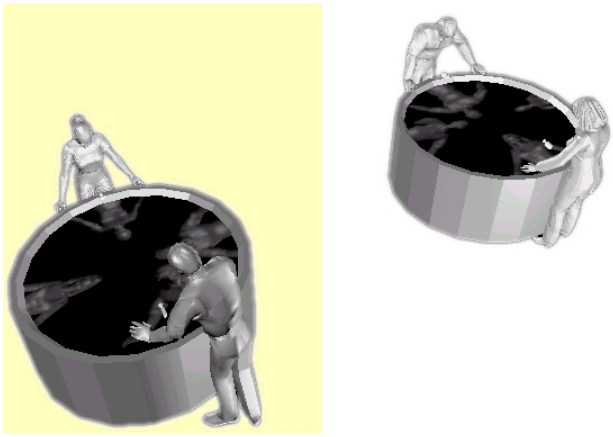


Figure 100. Le puits : idée générale

La projection horizontale présente un double intérêt. Non seulement les participants locaux peuvent se répartir en cercle autour du puits, mais les images des participants distants peuvent elles aussi être réparties de cette manière, superposées ou juxtaposées pour former l'image composée. Disposée horizontalement, la surface de projection perd bien évidemment son orientation verticale. Ce qui est affiché n'est donc plus perçu comme l'image directe des personnes, mais plutôt comme leur reflet probable sur la surface (Figure 101). Ces personnes ne nous semblent donc pas être "à l'envers", et la présence de ce reflet sans origine réelle provoque notre curiosité et nous pousse à vouloir comprendre ce que nous percevons.

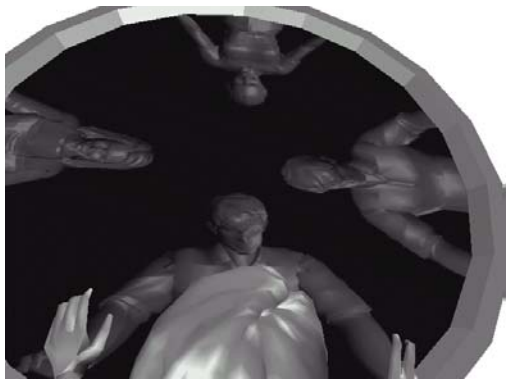


Figure 101. Le puits : vue subjective

En jouant sur la taille de la surface de projection, il est envisageable de relier plus de deux puits. Comme nous l'avons expliqué, la disposition horizontale des images et les possibilités de composition par superposition ou juxtaposition permettent en effet d'utiliser tout le périmètre du dispositif pour répartir les participants locaux et distants. On peut d'ailleurs remarquer que l'utilisation du dispositif reste la même quel que soit le nombre de puits et de participants impliqués : seule la répartition des personnes et des images autour du puits change.

2.4. Prototypage logiciel du système vidéo

La construction d'un prototype complet audio/vidéo du puits nécessite des moyens matériels assez conséquents. Il était donc important de pouvoir avoir une idée du fonctionnement du dispositif et des moyens nécessaires à sa réalisation avant de se lancer dans la conception et la réalisation de l'ensemble. Une modélisation du puits à l'aide de l'environnement de simulation et d'animation *Fabule* [Gas94] d'iMagis nous a permis d'avoir une vision assez réaliste de ce que nous voulions obtenir (Figure 100 et Figure 101). En enregistrant des séquences vidéo correspondant aux conditions de prise de vues envisagées (Figure 102), nous avons pu réaliser avec videoSpace et OpenGL un certain nombre de simulations de la composition des vues.

Figure 102. Image typique des conditions de prise de vue d'une des caméras d'un puits

Deux principaux modes de composition des images locales et distantes ont ainsi pu être étudiés. Le premier consiste à fusionner les images en les superposant par transparence (Figure 103, gauche). Le deuxième mode consiste à affecter à chaque source une région particulière de l'image composée, opérant ainsi une partition de celle-ci (Figure 103, droite). Chaque méthode présente des avantages et des inconvénients : la première, la fusion, produit une image plus homogène mais avec une faible résolution chromatique, tandis que la seconde, la partition, produit une image plus contrastée mais présentant des discontinuités.

Figure 103. Fusion et partition, deux exemples de composition de plusieurs sources d'images

Plusieurs variantes de ces deux modes ont pu être testées, en jouant sur la fonction de mélange utilisée pour la superposition ou sur l'orientation, l'échelle et le positionnement des images. Divers effets ont également pu être testés, comme la perturbation de l'image composée par des vagues similaires à celles de Liquid Views ou encore des compositions de type kaléidoscope.

2.5. Premiers prototypes matériels

Suite aux prototypes logiciels, un premier prototype matériel a été réalisé en deux exemplaires. Afin de simplifier la mise en œuvre de la projection, ce premier prototype est de forme carrée et non circulaire (Figure 104).

Figure 104. Les deux premiers prototypes

Un projecteur SVGA et un miroir sont placés dans chaque puits, tandis que deux caméras sont placées face à face, sur deux côtés de la surface de projection, à environ un mètre du sol. La surface de projection utilisée est un panneau de verre dépoli à l'acide. Le projecteur et les caméras de chaque puits sont reliés à une station SGI O2 placée à l'extérieur de celui-ci (Figure 105). Cette station est chargée de l'acquisition des images des deux caméras, de la transmission de ces images vers l'autre puits et de la composition et de la restitution de l'image composée. Les O2s ne permettant de traiter que deux sources vidéo simultanées nous avons utilisé ses entrées vidéo pour connecter deux caméras, l'une analogique (PAL) et l'autre numérique (SGI O2cam), et nous avons branché le projecteur sur la sortie vidéo informatique à la place du moniteur. Un serveur réalisé avec videoSpace se charge de l'acquisition des deux flux d'images sur chaque machine et de leur diffusion par VSMP. Plusieurs clients videoSpace utilisant OpenGL peuvent alors être utilisés sur chacune des deux machines pour composer les quatre sources d'images et afficher la composition sur la surface du puits.

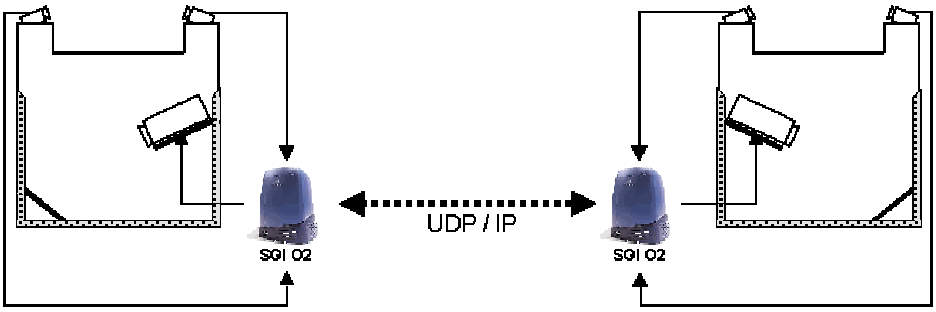


Figure 105. Montage vidéo pour la connexion des deux puits

Le système audio des prototypes a été conçu par le service acoustique du CSTB. La solution mise en œuvre utilise trois microphones cardioïdes suspendus au-dessus du centre du puits et trois haut-parleurs répartis uniformément sur son périmètre. La directivité des microphones utilisés et un couplage direct faible entre microphones et haut-parleurs permet d'éviter les phénomènes désagréables d'écho, de sifflement ou d'effet Larsen.

2.6. Evolution des prototypes

La construction de prototypes matériels nous a permis de vérifier certaines hypothèses et de découvrir de nouveaux problèmes et de nouvelles possibilités. Nous avions par exemple sous-estimé l'importance de l'éclairage de la pièce et de la nature du fond des prises de vues pour la qualité des images obtenues.

La réalisation d'un puits circulaire à partir du premier prototype ne devrait pas poser de problème. A priori, la seule différence est qu'une partie de l'image projetée sera masquée par les bords de la structure. La taille de l'image peut être augmentée au besoin en ajoutant une estrade autour et sous le puits, pour augmenter le recul du projecteur. Afin que la position penchée au-dessus du puits soit plus agréable, on peut également envisager d'ajouter des chaises hautes, tabourets ou accoudoirs. Une vitre pourrait enfin être placée au-dessus de la surface de projection pour la protéger et permettre de poser des objets sur le puits, comme des feuilles de papier, des verres ou des cacahuètes...

VideoSpace nous a permis de tester les prototypes sans nous soucier des transmissions de flux vidéo. Nous avons pu ainsi essayer plusieurs configurations comportant de une à quatre caméras par puits, certaines caméras étant simulées par des séquences enregistrées en raison des limitations du matériel d'acquisition des O2s. Suite à ces essais, une nouvelle évolution du prototype est en cours de réalisation. Cette évolution utilisera une mosaïque de composition vidéo analogique pour regrouper les images des différentes caméras d'un puits sur un unique flux vidéo. Le flux analogique ainsi créé pourra être envoyé vers le puits

distant par une unique liaison analogique ou un codec MPEG sur ATM et permettra en outre de n'utiliser qu'une seule entrée vidéo sur la machine distante pour pouvoir numériser l'ensemble des images d'un puits (figure).

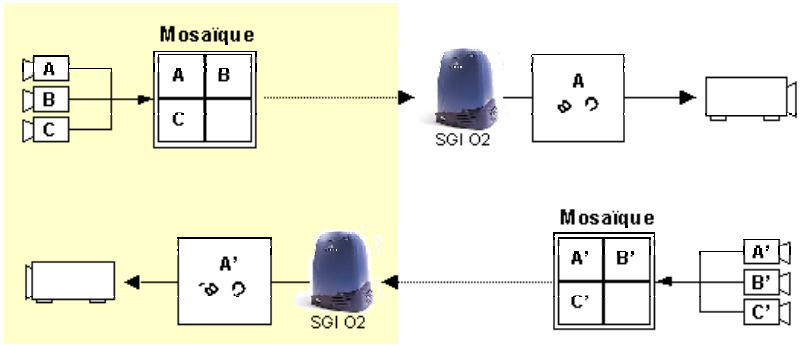


Figure 106. Montage vidéo de la future évolution du puits

Une nouvelle application utilisant videoSpace permet de créer des séquences de type mosaïque à partir de sources classiques (Figure 107). Ces séquences sont d'ors et déjà utilisées pour tester le logiciel qui sera nécessaire pour extraire les images de la mosaïque analogique et les composer.

Figure 107. Exemple de mosaïque vidéo

3. L'environnement informatique vu comme une source d'images

L'environnement X Window est l'un des rares environnements permettant en standard l'affichage d'une application sur une machine distante[56]. Cette possibilité d'affichage déporté se révèle particulièrement utile pour exécuter sur un serveur distant des applications nécessitant une grosse puissance de calcul, tout en contrôlant ces applications depuis une machine plus modeste. Comme nous l'avons expliqué précédemment, le protocole X ne transmet pas des images, mais les commandes graphiques de la Xlib nécessaires pour les produire. Ce choix est la conséquence

[56] A notre connaissance, seul OpenStep en était également capable

logique des capacités des réseaux et des applications utilisées à l'époque où X a été défini : la bande passante était faible, et les applications étaient avant tout textuelles.

Les capacités des réseaux ont beaucoup évolué. L'utilisation du système graphique également. Les librairies graphiques modernes comme OpenGL sont beaucoup plus complexes que la Xlib et nécessitent donc une plus grande puissance de calcul. Tel qu'il est implémenté par X ou GLX, l'extension OpenGL du protocole X, l'affichage déporté par transmission de commandes graphiques montre alors ses limites : la machine locale sur laquelle est affichée l'application doit elle aussi être puissante ou disposer de matériel spécialisé, comme une carte graphique accélérée, puisque c'est elle qui se charge de la production des images.

Depuis novembre 1999, SGI commercialise OpenGL Vizserver[57], une solution logicielle pour serveur Onyx2 qui redirige de façon transparente tous les appels OpenGL pour les exécuter localement, sur le serveur, puis transmet et affiche l'image ainsi produite sur une machine distante. Contrairement à l'approche X de l'affichage distant, Vizserver fonctionne selon un principe similaire à celui de RFB : les images sont calculées sur le serveur qui les compresse et les transmet ensuite au client qui se contente de les afficher. Les machines Onyx2 étant parmi les plus puissantes du moment, cette solution permet d'afficher à distance une application OpenGL tout en profitant à la fois des capacités stockage, de calcul et de traitement graphique du serveur.

Comme RFB, Vizserver tire partie de l'amélioration des performances du réseau pour décharger les applications clientes des traitements graphiques et leur fournir des images prêtes à afficher. Les applications existantes de ces deux systèmes sont pourtant très traditionnelles : elles se contentent d'afficher les images en un endroit unique et sans transformation. Notre troisième prototype de télépointeur vidéo montre qu'il est possible d'imaginer d'autres utilisations des flux d'images informatiques. Dans cette section, nous donnons trois autres exemples d'applications, deux collaboratives et l'une mono-utilisateur, qui soulignent l'intérêt de ce type de flux.

3.1. Utilisation de flux vidéo pour la téléconsultation cardiaque

Les applications de visualisation à but médical manipulent souvent plusieurs ensembles de données de natures diverses (échographie, IRM, modèle 3D, etc.) stockées de façon centralisée. La taille de ces données impose généralement l'utilisation de machines puissantes et disposant d'une grande capacité de mémoire pour

[57] http://www.sgi.com/software/vizserver/

pouvoir les manipuler et les présenter de manière interactive. Au-delà de ces besoins matériels pour un usage mono-utilisateur, la taille des données manipulées pose également un réel problème lorsque l'on envisage leur utilisation dans un contexte de téléconsultation. Non seulement les machines utilisées doivent être performantes, mais elles doivent également avoir accès aux données.

EchoCom est une application développée par GMD-FIT dans le cadre du projet CardiAssist[58] [Ber98]. Cette application est destinée à la téléconsultation cardiologique entre deux sites sur des images d'échographies. Les données utilisées décrivent généralement un unique cycle cardiaque, d'une durée approximative d'une seconde. L'acquisition d'un cycle se fait par échantillonnages échographiques successifs d'un volume 3D cubique d'environ 15cm de côté avec une précision inférieure au millimètre. Un enregistrement typique comporte ainsi une dizaine de volumes 200x200x200. Chaque donnée élémentaire du volume est un octet décrivant l'écho reçu par le dispositif d'acquisition : fort pour les tissus, faible pour les cavités. Un enregistrement typique nécessite donc 200x200x200x10x1 = 80 000 000 octets.

Chaque volume 3D de la séquence enregistrée est coupé selon un plan pour produire une animation montrant le cycle cardiaque complet. L'application présente cette animation ainsi qu'un modèle de cœur virtuel qui permet de situer le plan de coupe et sert de repère de référence (Figure 108). Plusieurs autres objets graphiques permettent également aux utilisateurs local et distant de contrôler de manière interactive l'orientation du plan de coupe.

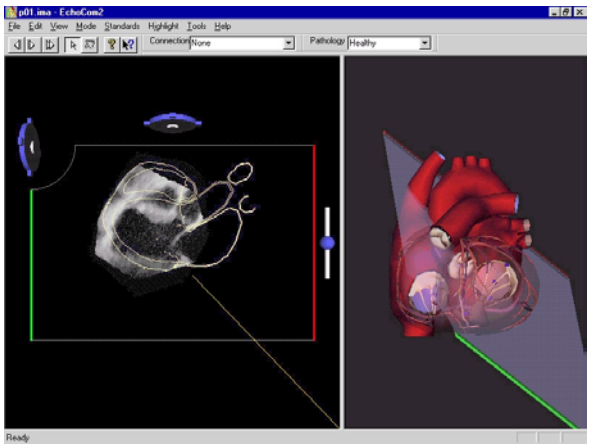


Figure 108. EchoCom (GMD, FIT.MMK)

La version actuelle d'EchoCom impose aux deux participants de disposer préalablement des données. Une fois lancées, les deux applications connectées chargent les données en mémoire et

[58] http://fit.gmd.de/hci/projects/cardi-assist/

produisent chacune les animations. Cette solution présente deux inconvénients :

- avant la téléconsultation, l'un des deux participants doit télécharger les données qui, comme nous l'avons vu, représentent plusieurs dizaines de mega-octets ;

- la machine utilisée pour exécuter l'application doit disposer d'une grande capacité mémoire pour charger les données et produire les animations[59].

Ces deux inconvénients font d'EchoCom l'exemple typique d'application qui pourrait bénéficier de l'utilisation de flux vidéo pour l'affichage d'une partie de son interface. Il est en effet assez facile d'imaginer qu'un serveur ayant accès aux données pourrait se charger de produire les animations et de les transmettre à deux applications qui se contenteraient de les afficher et transmettraient en retour les commandes nécessaires pour contrôler l'orientation du plan de coupe.

Afin de tester la validité de cette hypothèse, nous avons utilisé videoSpace pour réaliser un serveur capable de générer des animations de cycles cardiaques à partir des données EchoCom. Ce serveur diffuse les animations par le protocole VSMP et peut recevoir des commandes modifiant l'orientation du plan de coupe sur un autre canal UDP multicast. Une application cliente a ensuite été développée qui permet d'afficher les animations et de contrôler le plan de coupe à l'aide de touches du clavier. Le serveur a été réalisé en C++, tandis que deux versions du client ont été développées, l'une en C++ et l'autre en Java. VideoServer peut également servir de relais pour afficher les animations dans un navigateur Web (Figure 109).

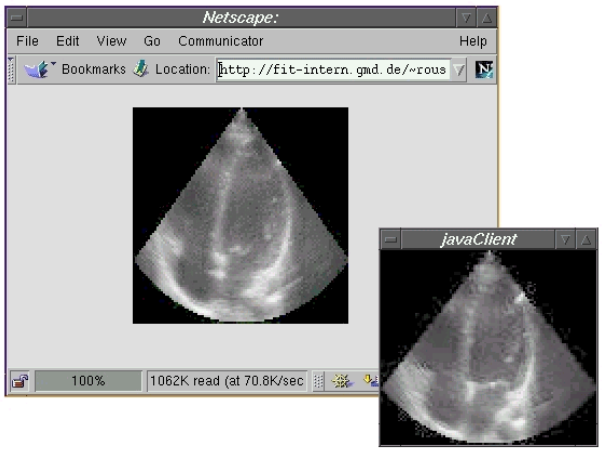


Figure 109. Animation d'un cycle cardiaque transmise par videoSpace vers un client Java et un navigateur Web

[59] Cette remarque est également valable pour une utilisation mono-utilisateur

En transformant la partie de l'application qui génère les animations en source d'images videoSpace, nous avons pu tester en quelques jours l'approche distribuée de la visualisation des données d'échographie. Cette nouvelle source nous permet également de tester de nouvelles formes d'interaction, comme l'usage de la main détourée par chroma-keying, et de construire d'autres prototypes montrant également l'image de l'interlocuteur distant.

3.2. Le ClearBoard du pauvre : une idée riche !

Pendant la mise au point du troisième prototype de télépointeur vidéo, nous avons eu l'occasion de tester plusieurs combinaisons d'affichage des images d'applications en cours d'exécution, de l'utilisateur et de ses mains détourées par chroma-keying. L'un de ces essais a donné naissance à videoWorkspace, un serveur HTTP écrit avec videoSpace qui envoie à ses clients par server-push la fusion par transparence des images d'une caméra placée devant un utilisateur et de son environnement de travail informatique (Figure 110).

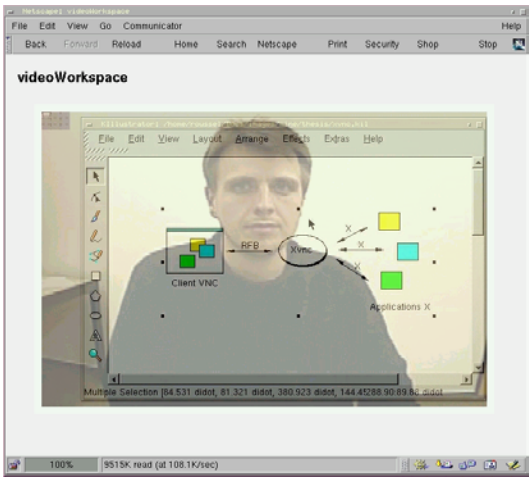


Figure 110. VideoWorkspace

L'image transmise par videoWorkspace ressemble à celles obtenues par F. Vernier et al. dans leurs interfaces augmentées par effet miroir [VLNC99]. Cependant, le but recherché n'est pas le même : les images de videoWorkspace ne sont pas destinées à l'utilisateur filmé, mais à d'autres personnes. VideoWorkspace est une sorte de "ClearBoard du pauvre"[60] : il permet à une personne distante de voir à la fois l'utilisateur et son environnement de travail, et ce depuis un simple navigateur Web. Il ne nécessite pas de matériel ou d'application spécifique pour celui qui regarde et permet à celui qui est filmé d'utiliser normalement ses applications pour montrer quelque chose à distance.

[60] A plus d'un titre ! L'écriture de videoWorkspace n'a pas coûté cher : le programme fait à peine une centaine de lignes de C++...

Les sources d'images correspondant à l'environnement informatique et à la caméra peuvent être spécifiées au lancement de videoWorkspace par des URLs telles que `rfb://mimix.gmd.de:1` et `vstp://gandalf.gmd.de/video`. Des serveurs VNC existant sur les plates-formes MS Windows, Macintosh, et X Window, le même exécutable de videoWorkspace peut donc fusionner des images provenant de la majorité des environnements actuels avec celles d'une caméra quelconque reliée à un videoServer.

VideoWorkspace n'est qu'un exemple de ce qui peut être obtenu rapidement en combinant un flux d'images d'environnement informatique avec d'autres flux de videoSpace. D'autres compositions seraient sans doute intéressantes. Nous pourrions par exemple ajouter l'image d'une caméra filmant les mains de l'utilisateur, avec ou sans chroma-keying. Au lieu d'utiliser un navigateur Web, nous pourrions également développer une application spécifique qui pourraient, comme notre troisième prototype de télépointeur vidéo, contrôler l'environnement informatique.

3.3. Prototypage de nouveaux environnements graphiques

Le modèle graphique des librairies actuelles comme OpenGL, DirectX ou Java3D est beaucoup plus évolué que celui des systèmes de fenêtrage que nous utilisons quotidiennement (MacOS, MS Windows ou X Window). Cette importante différence entre les possibilités graphiques accessibles à une application particulière et à celle qui gère les différentes fenêtres complique la mise en œuvre de techniques innovantes de gestion de documents ou d'applications. De récents travaux comme Elastic Windows [KS97], Mondrian [CDT99] ou Task gallery [RvDR+00] montrent pourtant qu'il y a encore de nombreuses possibilités à explorer dans ce domaine.

MacOS X, le futur système d'exploitation d'Apple, utilisera une nouvelle librairie graphique appelée Quartz. Quartz est basé sur le standard PDF (Portable Document Format) d'Adobe et offre des possibilités graphiques 2D avancées comme la gestion d'objets graphiques composés, l'antialiasing des caractères, la gestion de la transparence ou des transformations géométriques. Quartz est non seulement destiné au développement d'applications graphiques, mais il est également à la base d'Aqua, la partie du système responsable de la gestion du fenêtrage et des éléments d'interface de MacOS X[61]. Malheureusement, l'utilisation des possibilités graphiques de Quartz pour la gestion des documents et applications

[61] Pour plus de détails sur MacOS X, Quartz et Aqua, lire l'article qui leur est consacré sur le site d'Ars Technica, http://arstechnica.com/reviews/1q00/macos-x-gui/macos-x-gui-1.html

se limite pour le moment à quelques effets sans réel impact sur l'utilisateur (Figure 111).

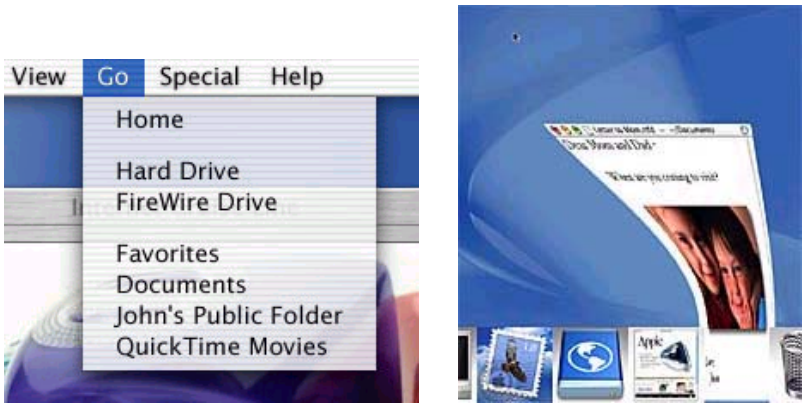


Figure 111. Utilisation de la transparence et des transformations géométriques dans MacOS X (Apple)

Les systèmes de fenêtrage des plates-formes MS Windows et MacOS sont intimement liés aux systèmes d'exploitation eux-mêmes et l'accès à leur fonctionnement interne ou leur modification sont difficiles. Ce n'est donc pas un hasard si Microsoft et Apple sont les seuls à pouvoir montrer des systèmes de fenêtrage pour ces plates-formes utilisant des transformations 3D (Task gallery, pour Microsoft) ou la transparence (Aqua/Quartz, pour Apple). Si ces exemples montrent la voie à suivre par l'utilisation de modèles graphiques plus évolués, les chances que la communauté de recherche puisse expérimenter de telles formes d'interaction avec les documents et applications sont encore faibles.

Notre troisième télépointeur vidéo et videoWorkspace fusionnent l'image de l'environnement informatique obtenue par un serveur VNC avec d'autres images vidéo. Dans ces deux exemples, l'image informatique est affichée sans transformation, mis à part l'utilisation de la transparence. D'autres traitements graphiques pourraient pourtant être utilisés pour transformer l'image ou en extraire des éléments. En particulier, en connaissant la position des différentes fenêtres dans l'image, il serait possible d'en extraire les sous-images correspondantes et de les réutiliser dans un autre contexte. Il serait ainsi possible de transmettre un flux vidéo correspondant à une unique application. Il serait également possible de réarranger les sous-images pour former une autre image de l'environnement informatique.

Nous avons commencé à explorer cette utilisation d'un flux d'images provenant d'un serveur VNC pour modifier l'apparence et l'interaction avec un environnement de travail informatique. La Figure 112 donne un exemple du type d'environnement que nous aimerions pouvoir expérimenter. Dans cet exemple, l'ordre d'affichage des fenêtres détermine leur transparence, leur taille et leur orientation. Aucune décoration n'est ajoutée aux fenêtres, mais une ombre permet d'en identifier les bords. Contrairement aux

effets d'Aqua, les transformations géométriques effectuées dans cet exemple ont un contenu sémantique : les deux fenêtres les plus récemment utilisées restent droites et opaques, tandis que les autres subissent une rotation et un léger changement de taille et sont rendues partiellement transparentes, se fondant ainsi dans l'arrière plan.

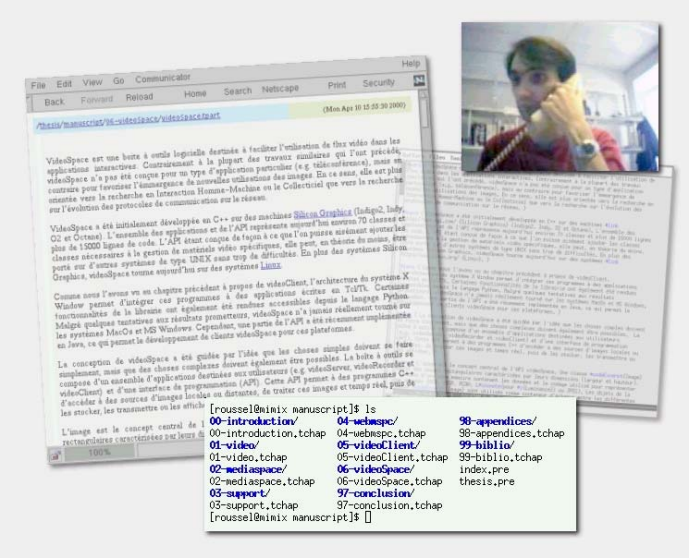


Figure 112. Exemple d'affichage envisagé pour un nouvel environnement graphique (simulation réalisée avec gimp)

Afin de pouvoir mettre en œuvre ce type d'environnements, nous avons développé un gestionnaire de fenêtres X Window minimaliste qui n'ajoute aucune décoration et place les applications les unes à côté des autres. Utilisé avec ce gestionnaire de fenêtres, un serveur Xvnc configuré pour un écran virtuel rectangulaire produit des images semblables à celle de la Figure 113. Notre application connaissant la position et la taille des différentes fenêtres présentes dans les images du serveur VNC, elle peut alors utiliser le placage de texture d'OpenGL pour transformer et afficher chaque fenêtre indépendamment des autres et composer ainsi le nouvel environnement.

Figure 113. Juxtaposition de fenêtres non décorées

Notre prototype actuel affiche les différentes fenêtres en faisant varier leur transparence et en les pivotant, mais se contente d'ombre simpliste à une couleur. Le résultat obtenu n'est pas encore celui présenté par la Figure 112, mais l'amélioration de la technique d'ombrage devrait permettre de nous en rapprocher. Si l'affichage n'est pas encore parfait, il est cependant actif : toute modification du contenu d'une fenêtre par une application est répercutée dans l'affichage de notre prototype. Une transformation des coordonnées de la souris serait nécessaire pour pouvoir transmettre des événements au serveur VNC. Cette transformation n'est pas effectuée pour le moment mais ne devrait pas poser de problème particulier si l'on se limite à des transformations simples. En attendant, un client VNC classique peut être utilisé pour interagir avec les applications et voir les changements dans notre prototype.

L'utilisation de VNC et de videoSpace devrait permettre de prototyper différentes transformations graphiques de l'affichage des applications. VNC permet en outre d'envisager l'utilisation de cette solution sur des images provenant de systèmes MS Window ou MacOS. Il est cependant encore trop tôt pour dire si les performances seront à la hauteur de nos ambitions. Sur la plate-forme X Window, il existe au moins une alternative à l'utilisation de VNC pour obtenir l'image de départ. Xvfb est un autre serveur X qui lui aussi utilise une zone mémoire comme zone d'affichage. A la différence de Xvnc cependant, Xvfb peut placer cette zone dans un segment de mémoire partagée, ce qui permettrait à notre gestionnaire de fenêtres d'accéder d'y accéder directement, sans passer par le réseau.

4. Résumé du chapitre

Nous avons décrit dans ce chapitre trois contextes d'utilisation de la vidéo très différents mais qui illustrent bien les possibilités de prototypage de nouveaux usages offertes par videoSpace.

Nous avons présenté une nouvelle approche au problème de la communication gestuelle dans les collecticiels basée sur l'utilisation de l'image des mains des participants détourée par chroma-keying comme télépointeur. Nous avons décrit trois prototypes réalisés successivement avec videoSpace ainsi que différentes simulations qui nous ont permis de valider cette approche et d'en mesurer les limites.

Nous avons présenté *le puits*, un dispositif de *téléconvivialité* qui permet à des groupes de personnes distantes de communiquer de façon informelle lors des pauses, avant ou après des téléréunions ou visioconférences. Nous avons décrit les aspects matériels et logiciels de ce dispositif ainsi que le rôle important joué par videoSpace lors de sa conception et de son prototypage.

Nous avons enfin présenté plusieurs autres travaux en cours qui reposent sur l'utilisation de flux vidéo générés par l'environnement informatique. Bien qu'encore à l'état d'ébauche, ces travaux montrent l'intérêt que peuvent avoir des flux d'images synthétisées dans le cadre d'applications collaboratives et du prototypage de nouveaux environnements graphiques et laissent entrevoir des perspectives intéressantes de développements ultérieurs.

Conclusion

Résumé de la contribution

L'objet de cette thèse est l'étude de l'intégration de la vidéo dans les systèmes informatiques interactifs. Notre travail est motivé par une double volonté : nous voulons comprendre le fonctionnement des systèmes existants pour pouvoir reproduire leurs usages, et nous voulons également poser les bases logicielles qui permettront d'en développer de nouveaux.

L'étude de différents systèmes commerciaux et prototypes de recherche nous a permis de faire ressortir quatre grandes fonctions de la vidéo dans les systèmes informatiques interactifs : la communication formelle, la communication informelle, le partage de vues dans des activités collaboratives et l'interaction homme-machine par le traitement informatique des images. Ces quatre fonctions ne sont nullement exclusives, certains systèmes les combinant pour diversifier les usages.

Nous avons choisi d'étudier plus en détail une famille particulière de systèmes, les mediaspaces, en raison de la diversité des usages qu'ils permettent et de la singularité de leur mode de développement. Cette étude détaillée nous a permis de dégager trois dimensions essentielles qui caractérisent ces systèmes :

- leur *intégration* dans les habitudes des utilisateurs, qui offre à ces derniers un moyen d'accès permanent, rapide et intuitif aux autres personnes ;
- leur *flexibilité*, qui permet à chacun d'adapter le système à ses besoins particuliers ;
- leur *intimité*, ou la facilité avec laquelle les utilisateurs comprennent, utilisent et font confiance aux mécanismes que le système leur propose pour protéger leur vie privée.

Nous avons montré que ces trois propriétés peuvent servir de guide à la conception de l'infrastructure logicielle de nouveaux environnements de communication vidéo. Nous avons décrit comment les langages et protocoles sur lesquels repose le Web peuvent être utilisés pour mettre en œuvre une telle infrastructure et nous avons démontré la faisabilité de cette mise en œuvre en réalisant Mediascape et videoServer, deux serveurs HTTP dédiés au contrôle d'une infrastructure audio/vidéo analogique et à la transmission de flux vidéo numériques.

Mediascape et videoServer permettent à leurs utilisateurs d'intégrer des services de communication vidéo au sein de leurs documents HTML. Nous avons montré comment cette intégration offre aux membres d'un groupe un accès rapide et personnalisable à chaque individu ainsi qu'à l'ensemble du groupe. Nous avons décrit les usages informels que cette approche autorise ainsi que les mécanismes de contrôle et de notification de videoServer qui permettent aux utilisateurs de trouver un compromis entre l'accessibilité permanente et la protection de leur vie privée. Nous avons par ailleurs montré que l'intérêt de ces mécanismes ne se limite pas aux usages de la vidéo mais qu'ils peuvent se généraliser à l'échange de tous types de documents entre individus.

Nous avons montré comment les possibilités d'intégration de la vidéo offertes par videoServer peuvent être étendues en y accédant non plus depuis des navigateurs ou autres applications standards du Web, mais depuis des clients dédiés et à travers des protocoles spécifiques. Nous avons montré qu'en plus des usages informels disponibles à travers le Web, ce type de client vidéo spécifique permet d'augmenter des applications existantes telles qu'un éditeur partagé ou d'en développer de nouvelles pour passer progressivement à des situations plus formelles.

En nous basant sur les développements logiciels de videoServer et des clients et protocoles associés, nous avons conçu videoSpace, une boîte à outils logicielle destinée à faciliter l'intégration de la vidéo dans les systèmes informatiques interactifs. VideoSpace reprend les caractéristiques de Mediascape et videoServer qui permettent l'intégration de flux vidéo dans les documents et y ajoute une interface de programmation qui permet aux concepteurs d'applications de bénéficier également de ces flux d'images.

Nous avons décrit l'interface de programmation de videoSpace et montré à quel point elle permet de faire abstraction des détails techniques d'acquisition, de codage ou de transmission des images pour se concentrer uniquement sur leur utilisation. Nous avons enfin présenté trois séries de prototypes qui illustrent les possibilités offertes par notre boîte à outils pour la conception et l'expérimentation de nouveaux usages de la vidéo.

Limites et perspectives

L'intégration de la vidéo dans l'environnement informatique par les documents et le Web a prouvé son intérêt pour des usages informels de la vidéo par les témoignages des utilisateurs de videoServer. Ceci n'est pas surprenant. Les qualificatifs que nous pourrions choisir pour décrire l'usage quotidien du Web seraient en effet très proches de ceux qui décrivent la communication informelle : fréquent, bref, intermittent, impromptu, sans réel début ou fin.

Le *puits* se place lui aussi dans un contexte de communication informelle, tandis que l'usage de la main comme télépointeur et videoWorkspace peuvent être mis dans la catégorie du partage de vues dans les activités collaboratives. Les deux fonctions de la vidéo que nous avons le moins abordées dans cette thèse sont la communication formelle et le traitement d'image pour l'interaction homme-machine. Nous ne touchons à la première que par l'utilisation combinée du téléphone ou d'applications audio avec videoServer, tandis que nous effleurons la seconde avec quelques démonstrateurs basés sur des différences entre images successives.

VideoSpace devrait permettre à l'avenir de développer des applications destinées à des usages de la vidéo plus formels. La prise en charge de l'audio semble nécessaire à ce type d'usages, et outre les traditionnels problèmes de synchronisation des flux et de gestion de session, on peut penser que l'intégration du son va poser quelques questions relatives aux usages. Il serait intéressant d'étudier les interfaces nécessaires et les mécanismes de contrôle et de notification à prévoir pour que l'audio soit aussi accessible que la vidéo sans être pour autant intrusive.

VideoSpace devrait également nous permettre de développer ou de réutiliser des filtres de traitement d'images existants pour l'interaction homme-machine. Une fois ces filtres implémentés, nous pourrions alors une nouvelle fois nous concentrer sur les usages : que peut-on faire si l'on est capable de localiser, d'identifier et de suivre des objets présents à l'image ?

Les perspectives offertes par notre travail résident principalement dans l'utilisation future de videoSpace pour expérimenter de nouveaux usages de la vidéo. Nous pensons d'ores et déjà que le modèle d'événement et de flot d'exécution sur lequel repose les multiplexeurs de videoSpace sera sans doute insuffisant face à la complexité de certaines applications interactives. D'un autre côté, nous avons montré que l'utilisation de flux vidéo générés par l'environnement informatique ouvre des possibilités intéressantes pour la conception de nouveaux environnements graphiques.

Nous pensons que l'évolution de videoSpace doit s'inscrire dans un mouvement plus ambitieux de conception d'une infrastructure

logicielle globale et non limitée à la vidéo qui permettra aux concepteurs de logiciels de développer des applications plus adaptées à leurs utilisateurs et aux usages émergents de l'informatique.

Annexe A – Acquisition vidéo en temps réel

Les performances des matériels d'acquisition vidéo n'ont cessé de s'améliorer au cours des dix dernières années. On peut aujourd'hui trouver pour un prix raisonnable des caméras que l'on branche directement sur le port USB d'un ordinateur personnel et qui sont capables de numériser en temps réel des flux vidéo dont la résolution peut aller jusqu'à 640x480 pixels. Accompagnant l'évolution des matériels, diverses interfaces de programmation ont été développées pour permettre leur utilisation sur la plupart des plates-formes actuelles. On peut ainsi citer QuickTime[62] d'Apple, Video For Windows ou plus récemment DirectShow[63] de Microsoft ou les Digital Media Libraries[64] de SGI.

Les interfaces de programmation existantes pour la gestion du matériel d'acquisition vidéo sont totalement incompatibles entre elles. En août 1999, Sun et IBM ont annoncé la version 2.0 de Java Media Framework[65], une interface de programmation qui permet d'intégrer l'audio et la vidéo dans des applications Java. Une telle librairie pourrait faciliter dans le futur la portabilité des applications vidéo. Cependant, près d'un an après son annonce, l'accès au matériel vidéo par Java Media Framework n'est aujourd'hui réellement implémenté que sur les plates-formes MS Windows et Sun Solaris...

La boîte à outils videoSpace ainsi que les différents prototypes présentés dans cette thèse ont été développés sur des machines SGI sous IRIX et des PCs sous Linux. Le code que nous avons

[62] http://developer.apple.com/quicktime/

[63] http://www.microsoft.com/DirectX/dxm/help/ds/

[64] http://www.sgi.com/developers/devtools/apis/digitalmedia.html

[65] http://java.sun.com/products/java-media/jmf/

écrit peut être compilé puis exécuté sans aucune modification sur l'une ou l'autre de ces deux plates-formes. Nous décrivons dans cette Annexe quelques particularités des interfaces de programmation vidéo bas niveau disponibles sous IRIX et Linux à partir desquelles nous avons du travailler.

1. Acquisition vidéo sous SGI IRIX

La programmation du matériel vidéo SGI se fait par l'utilisation des Digital Media Libraries. Au fil des années, ces librairies ont été modifiées par SGI pour prendre en compte leurs nouveaux modèles de machines et de carte d'acquisition ainsi que l'évolution de leur système d'exploitation IRIX. Toutefois, contrairement à d'autres concepteurs d'interfaces de programmation, SGI a su limiter les modifications effectuées de sorte qu'il est encore possible aujourd'hui de concevoir une unique application d'acquisition vidéo destinée à la fois à une machine d'il y a six ans et au dernier modèle sorti. VideoServer a ainsi pu être compilé et exécuté sur des machines de type Indy, Indigo2, O2 ou Octane tournant sous IRIX 5.2, 6.2, 6.3, 6.4 et 6.5.

Le nombre de flux vidéo pouvant être traités simultanément sur ces machines est limité à la fois par leurs entrées/sorties physiques - le nombre de prises vidéo - mais également par leur architecture interne. Bien que certains modèles disposent de plusieurs entrées et d'une sortie vidéo, la plupart ne permettent de gérer qu'au plus deux flux simultanés. Cette limite est liée à l'utilisation de zones mémoire particulières pour la manipulation des images. Ainsi, sur une O2 équipée d'une carte vidéo avec deux entrées et une sortie, les possibilités d'acquisition et de production d'images sont les suivantes :

- acquisition d'une source ;
- double acquisition d'une source unique ;
- acquisition simultanée de deux sources différentes ;
- production d'un flux sur la sortie vidéo ;
- acquisition d'une source et production simultanée d'un flux sur la sortie.

Cette restriction des possibilités d'acquisition et de production sur une machine qui est pourtant nettement supérieure à un PC ordinaire illustre bien les limites des systèmes actuels pour l'utilisation de la vidéo en temps réel. Un flux vidéo entrant ou sortant directement de la machine est une ressource disponible en quantité limitée.

Les Digital Media Libraries permettent à une application de verrouiller les zones mémoire utilisées pour l'acquisition ou la production de flux vidéo. Si ces zones ne sont pas verrouillées, elles peuvent être réquisitionnées par une autre application à tout moment, ce qui a comme conséquence l'interruption immédiate de l'acquisition ou de la production d'images de la première application. Nous avons utilisé cette possibilité pour permettre une gestion coopérative des ressources vidéo par nos différentes applications.

En termes de performance, les dernières machines SGI sont capables d'acquérir ou de produire des flux vidéo au format PAL (768x576 pixels à 25 images/seconde) sans aucun problème. Certains modèles, comme l'O2, disposent en outre d'un microprocesseur dédié à la compression/décompression JPEG et sont donc capables de coder et décoder des flux vidéo en temps réel sans utiliser le microprocesseur principal.

2. Acquisition vidéo sous Linux

Depuis la version 2.2 du noyau, Linux intègre Video For Linux[66], une interface de programmation unique pour tous les matériels vidéo. Cette interface de programmation est relativement simple mais incroyablement mal spécifiée. L'écriture d'une application vidéo ou d'un pilote (driver) pour un nouveau matériel est une tâche relativement complexe en raison des nombreuses interprétations contradictoires de la spécification. A titre d'exemple, s'il existe un unique format VIDEO_PALETTE_RGB24 qui décrit un pixel sur trois octets, les concepteurs et utilisateurs de Video For Linux n'arrivent pas à se mettre d'accord sur l'ordre des trois composantes de ce format (RGB ou BGR ?).

Les pilotes disponibles pour les matériels d'acquisition vidéo sont pour la plupart encore en cours de développement. Le développement se faisant dans bien des cas sans l'assistance du fabricant, la qualité du logiciel et les performances obtenues sont extrêmement variables. Néanmoins, quelques matériels tels que ceux basés sur les composants de marque Brooktree ou de type CPiA[67] permettent de numériser des flux vidéo au format PAL ou dans un format s'en approchant (640x480 à 30 images secondes).

Video For Linux ne permet pas l'acquisition multiple d'une source. La librairie n'offre aucun moyen standard de réquisition de ressource comparable à celui des Digital Media

[66] http://roadrunner.swansea.uk.linux.org/v4l.shtml

[67] Voir http://www.metzlerbros.de/bttv.html et http://webcam.sourceforge.net/ pour plus d'informations sur ces composants

Libraries de SGI. Pour reproduire la possibilité de gestion coopérative des ressources vidéo par les applications, nous avons utilisé dans videoSpace un canal de signalisation implémenté par UDP multicast. Grâce à ce canal commun à toutes les applications videoSpace, une application peut réclamer une ressource en cours d'utilisation.

Un certain nombre de propositions ont été faites pour améliorer Video For Linux. Video For Linux 2[68] est sans aucun doute l'une des plus intéressantes, proposant une spécification très complète et très précise et permettant en outre des accès multiples aux ressources vidéo. L'adoption de cette nouvelle interface de programmation nécessite toutefois la réécriture des pilotes existants, ce qui constitue un frein non négligeable à son adoption. En mars 2000, SGI a également annoncé la disponibilité prochaine des Digital Media Libraries pour Linux[69]. Bien qu'il soit encore trop tôt pour se prononcer sur l'avenir de cette nouvelle interface de programmation, on peut y voir un espoir de convergence des années d'expérience de SGI en matière de vidéo numérique et des capacités de développement de la communauté Linux.

[68] http://millennium.diads.com/bdirks/v4l2.htm

[69] http://www.sgi.com/newsroom/press_releases/2000/march/digimedia_linux.html

Annexe B – miniserver.cxx

```
#include <videoSpace/base/CommandLineUtils.H>
#include <videoSpace/imageSource/ImageSource.H>
#include <videoSpace/network/TcpUtils.H>
#include <videoSpace/network/HttpMessage.H>
#include <videoSpace/imageSink/network/ServerPushImageSink.H>

using namespace videoSpace ;

// Délare la procédure exécutée pour chaque connexion au serveur
// (le code de cette procédure figure à la fin du fichier)
void serveVideo(char *src, SimpleTcpConnection *client) ;

int
main(int argc, char **argv) {
  int PORT = 5556 ;
  char *SOURCE = "vsmp://localhost:5557" ;

  // Traite les arguments de la ligne de commande pour déterminer
  // le numéro de port TCP à utiliser et la source vidéo à servir
  if (parseCommandLine(argc, argv, "p:i:", "is", &PORT, &SOURCE)<0){
    std::cerr << std::endl << argv[0] << " [-p port] [-i source]" ;
    std::cerr << std::endl ;
    exit(1) ;
  }

  // Crée un serveur TCP
  SimpleTcpServer server(PORT) ;

  // Répète à l'infini…
  for (;;) {

    // Attend qu'un client se présente
    SimpleTcpConnection *client = server.waitForNewClient() ;

    int fd = client->getFd() ;
    HttpMessage msg ;

    try {

      // Décode le message HTTP envoyé par le client
      if (msg.parseFromStream(fd)) {
```

```
        // Affiche le nom de la machine distante
        std::cout << "(" << client->machineLookUp() << ", " ;
        //    ainsi que le nom de l'utilisateur distant (s'il est connu)
        std::cout << client->userLookUp() << ") " ;
        //    et la première ligne de la requête HTTP
        std::cout << msg.startLine() << std::endl ;

        // Envoie au client les images de la source spécifiée
        serveVideo(SOURCE, client) ;
      }
    } catch (Error e) {
      std::cerr << std::endl << "ERROR: " << e.message << std::endl ;
    }

    delete client ;
  }
}

void
serveVideo(char *src, SimpleTcpConnection *client) {

  // Descripteur utilisé pour savoir si le client a fermé la connexion HTTP
  FileDescriptor signal(client->getFd(), Multiplexing::READABLE) ;

  // Crée un objet source à partir de l'URL spécifiée
  ImageSource *source = createImageSource(Image::JPEG, src) ;

  // Crée un diffuseur d'image par server-push sur la connexion HTTP
  ServerPushImageSink streamer(client->getFd()) ;

  // Crée un multiplexeur puis lui associe les trois objets précédents
  Multiplexer mux ;
  mux.addObject(signal) ;
  mux.addObject(*source) ;
  mux.addObject(streamer) ;

  Image img ;

  // Active la source et le diffuseur et répète tant qu'ils sont actifs…
  for (source->start(), streamer.start();
       source->isActive() && streamer.isActive();) {

    // Attend que des données arrivent de la source ou par la connexion HTTP
    // ou que le diffuseur soit prêt
    mux.reset() ;
    mux.multiplex() ;

    // Si quelque chose est arrivé sur la connexion HTTP, stoppe la transmission
    if (signal.getState()&Multiplexing::READABLE) break ;

    // Si une nouvelle image est disponible, demande au diffuseur de s'en occuper
    if (source->getNextImage(&img)) streamer.handle(&img) ;
  }

  delete source ;
}
```

Annexe C – videoClient.cxx

```
#include <videoSpace/base/CommandLineUtils.H>
#include <videoSpace/base/SignalUtils.H>
#include <videoSpace/imageSource/ImageSource.H>
#include <videoSpace/imageFilter/Convolution.H>
#include <videoSpace/imageFilter/Difference.H>
#include <videoSpace/imageFilter/Scaling.H>
#include <videoSpace/imageSink/ImageSink.H>

using namespace videoSpace ;

// Booléen utilisé pour savoir si le programme a été interrompu
// par l'envoi d'un signal
bool signaled=false ;
void exitProc(int) { signaled = true ; }

int
main(int argc, char **argv) {
  char *SOURCE = 0 ;
  std::string FILTER = "None" ;
  char *SINK = 0 ;

  // Traite les arguments de la ligne de commande pour en extraire
  // l'URL de la source d'image, le nom éventuel du filtre à appliquer
  // et l'URL du visualisateur/enregsitreur/diffuseur

  if (parseCommandLine(argc, argv,
      "i:f:o:", "sSs", &SOURCE, &FILTER, &SINK)<0) {
    std::cerr << std::endl << argv[0] ;
    std::cerr << " [-i source] [-f filter] [-o viewer_recorder_netsource]" ;
    std::cerr << std::endl ;
    exit(1) ;
  }

  // Par défaut, la source est la caméra locale…
  if (!SOURCE) SOURCE = "videoin:/anydev/anynode" ;

  // … et le flux vidéo est affiché à l'écran
  if (!SINK) SINK = " gtkwindow://localhost:0?title=videoClient" ;

  try {
    Multiplexer mux ;
```

```
    // Crée la source puis l'ajoute aux objets gérés par le multiplexeur
    ImageSource *source = createImageSource(Image::RGB,SOURCE) ;
    mux.addObject(*source) ;

    Image img ;

    // Crée le filtre correspondant au traitement demandé
    SimpleFilter *filter=0 ;
    if (FILTER=="edges")
        filter = (SimpleFilter *) new Convolution_3x3(0,-1,0, -1,4,-1,0, -1,0,
0,1) ;
    else if (FILTER=="blur")
        filter = (SimpleFilter *) new Convolution_3x3(1,1,1, 1,1,1, 1,1,1, 0,9) ;
    else if (FILTER=="boost")
        filter = (SimpleFilter *) new Convolution_3x3(-1,-1,-1,-1,21.6,-1,-1,-1,-1,
0,9) ;
    else if (FILTER=="emboss")
        filter = (SimpleFilter *) new Convolution_3x3(-1,-1,0, -1,0,1, 0,1,1,
120,1) ;
    else if (FILTER=="difference")
        filter = (SimpleFilter *) new ImageDifference ;
    else if (FILTER=="nobackground")
        filter = (SimpleFilter *) new ImageDifference(25,1) ;
    else if (!strncmp(FILTER.c_str(),"resize-",7))
        filter = (SimpleFilter *) new ResizeFilter(FILTER.c_str()+7) ;

    // Crée le visualisateur/enregistreur/diffuseur puis l'ajoute aux objets
    // gérés par le multiplexeur
    ImageSink *sink = createImageSink(SINK) ;
    mux.addObject(*sink) ;

    // Si un signal est reçu, le programme doit être stoppé
    trapAllSignals((SIG_PF)exitProc) ;

    // Active la source et le visualisateur/enregistreur/diffuseur
    source->start() ;
    sink->start() ;

    // Tant que les deux sont actifs et qu'aucun signal n'a été reçu…
    while (!signaled && source->isActive() && sink->isActive()) {
        // Attend que l'un des deux objets soit prêt
        mux.reset() ;
        mux.multiplex() ;

        // S'il y a une nouvelle image…
        if (source->getNextImage(&img)) {
          // Applique le filtre éventuel
          if (filter) filter->apply(&img) ;
          // Envoie l'image au visualisateur/enregistreur/diffuseur
          sink->handle(&img) ;
        }
    }

    delete source ;
    delete sink ;

  } catch (Error e) {
    std::cerr << "ERROR! " << e.message << std::endl ;
  }

}
```

Bibliographie

[AH94] A. Adler and A. Henderson. "*A room of our own: experiences from a direct office share*". In CHI '94. Conference proceedings on Human factors in computing systems, pages 138-144. ACM, New York, 1994. http://www.acm.org/pubs/citations/proceedings/chi/191666/p138-adler/

[Azu97] R. Azuma. "*A survey of augmented reality*". Presence: Teleoperators and Virtual Environments, 6(4):355-385, August 1997. http://www.cs.unc.edu/~azuma/Arpresence.pdf

[Bea00] M. Beaudouin-Lafon, "*Ceci n'est pas un ordinateur - Perspectives sur l'Interaction Homme-Machine*", Numéro spécial "Informatiques - enjeux, tendances, volutions", sous la direction de René Jacquart.Technique et Science Informatique, TSI 19(1-2-3), janvier 2000, pp 69-74.

[BD95] R. Bentley and P. Dourish. "*Medium versus mechanism: Supporting collaboration through customization*". In Proceedings of European Conference on Computer-Supported Cooperative Work ECSCW'95, Stockholm, pages 133-148. Kluwer Academic, September 1995. ftp://ftp.parc.xerox.com/pub/dourish/ecscw95-customisation.ps

[Ber98] T. Berlage. "*Augmented-reality communication for diagnostic tasks in cardiology*". IEEE Transactions on Information Technology in Biomedicine, 2(3):169-173, 1998. http://zeus.gmd.de/~berlage/augreal-titb.pdf

[BFK+94] V. Bellotti, R. Fish, R. Kraut, P. Dourish, B. Gaver, A. Adler, S. Bly, M. Mantei, and G. Moore. "*Debating the Media Space design space*". In Panels' section of the Conference Companion of ACM CHI'94 Conference on Human Factors in Computing Systems, pages 193-194. ACM, New York, April 1994. http://www.acm.org/pubs/citations/proceedings/chi/259963/p193-bellotti/

[BG96] T. Barba and P-O. Giffard. "*Les téléconférences multimédias*". Hermès, 1996.

[BHI93] S.A. Bly, S.R. Harrison, and S. Irwin. "*Mediaspaces: Bringing people together in a video, audio and computing environment*". Communications of the ACM, 36(1):28-47, January 1993. http://www.acm.org/pubs/citations/journals/cacm/1993-36-1/p28-bly/

[BLCL+94] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, and A. Secret. "*The World Wide Web*". Communications of the ACM, 37(8):76-82, August 1994. http://www.acm.org/pubs/citations/journals/cacm/1994-37-8/p76-berners-lee/

[BLFM98] T. Berners-Lee, R. Fielding, and L. Masinter. "*Uniform Resource Identifiers (URI): Generic Syntax*". Draft standard, RFC 2396, IETF, August 1998. http://www.pasteur.fr/infosci/RFC/23xx/2396

[BM86] W. Buxton and B. Myers. "*A study in two-handed input*". In CHI '86. Conference proceedings on Human factors in computing systems, pages 321-326. ACM, New York, April 1986. http://www.acm.org/pubs/articles/proceedings/chi/22627/p321-buxton/p321-buxton.pdf

[BM90] W. Buxton and T. Moran. "*EuroPARC's Integrated Interactive Intermedia Facility (IIIF): Early Experiences*". In Multi-User Interfaces and Applications, pages 11-34. S. Gibbs and A.A. Verrijn-Stuart, North-Holland, September 1990. Proceedings of IFIP WG8.4 Conference, Heraklion, Greece. http://www.dgp.toronto.edu/OTP/papers/bill.buxton/iiif.html

[Bér99] F. Bérard. "*Vision par ordinateur pour l'interaction homme-machine fortement couplée*". Thèse de nouveau doctorat, Université Joseph Fourier, Grenoble (France), Novembre 1999. http://iihm.imag.fr/publs/1999/THESE1999_Berard.pdf

[BT91] A. Borning and M. Travers. "*Two approaches to casual interaction over computer and video networks*". In Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, pages 13-20. ACM, New York, 1991. http://www.acm.org/pubs/citations/proceedings/chi/108844/p13-borning/

[Bux97] W. Buxton. "*Living in Augmented Reality: Ubiquitous Media and Reactive Environments*". In K.E. Finn, A.J. Sellen, and S.B. Wilbur, editors, Video Mediated Communication. Lawrence Erlbaum Associates, 1997. http://www.dgp.toronto.edu/OTP/papers/bill.buxton/augmentedReality.html

[Cad94] C. Cadoz. "*Le geste canal de communication homme/machine*". La communication "instrumentale". Technique et Science Informatique, 13(1):31-61, 1994.

[CBCC98] J. Coutaz, F. Bérard, E. Carraux, and J. Crowley. "*Early experience with the mediaspace CoMedi*". In IFIP Working Conference on Engineering for Human-Computer Interaction, Heraklion, Crete, 1998. ftp://ftp.imag.fr/imag/IIHM/ENGLISH/publications/1998/EHCI98_CoMedi.pdf

[CCB00] J.L. Crowley, J. Coutaz, and F. Bérard. "*Perceptual user interfaces: things that see*". Communications of the ACM, 43(3):54-64, March 2000. http://www.acm.org/pubs/citations/journals/cacm/2000-43-3/p54-crowley/

[CDFN95] P. Curtis, M. Dixon, R. Frederick, and D.A. Nichols. "*The Jupiter audio/video architecture: secure multimedia in network places*". In Proceedings of the third international conference on Multimedia '95, pages 79-90. ACM, New York, 1995. http://www.acm.org/pubs/citations/proceedings/multimedia/217279/p79-curtis/

[CDT99] P. Carlier, A. Derycke, and J-C. Tarby. "*Mondrian, un agent d'interface pour accompagner les collecticiels*". In Actes des onzièmes journées francophones sur l'Interaction Homme Machine (IHM'99), Montpellier, Novembre 1999.

[CFBS97] J. Cooperstock, S. Fels, W. Buxton, and K.C. Smith. "*Reactive environments: Throwing away your keyboard and mouse*". Communications of the ACM, 40(9):65-73, September 1997. http://www.acm.org/pubs/citations/journals/cacm/1997-40-9/p65-cooperstock/

[CFKL92] C. Cool, R.S. Fish, R.E. Kraut, and C.M. Lowery. "*Iterative Design of Video Communication Systems*". In Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work, Toronto, Ontario, pages 25-32. ACM, New York, November 1992. http://www.acm.org/pubs/citations/proceedings/cscw/143457/p25-cool/

[Cha94] S. Chatty. "*Extending a graphical toolkit for two-handed interaction*". In Proceedings of the ACM symposium on User interface software and technology, pages 195-204. ACM, New York, November 1994. http://www.acm.org/pubs/citations/proceedings/uist/192426/p195-chatty/

[Coc00] Patrick Cocquet. "*L'avènement d'un protocole universel*". La Recherche, numéro spécial Internet, février 2000.

[Con97] S. Conversy. "*Sons d'environnement naturel pour le suivi d'activités d'arrière-plan*". In Actes des neuvièmes journées francophones sur l'Interaction Homme Machine (IHM'97), Futuroscope, Septembre 1997. http://www-ihm.lri.fr/~conversy/Papers/ihm97.ps.gz

[Cur99] Hubert Curien. "*Découverte et innovation*". Le Monde, 27 octobre 1999. http://www.lemonde.fr/article/0,2320,28275,00.html

[DABH96] P. Dourish, A. Adler, V. Bellotti, and A. Henderson. "*Your Place or Mine ? Learning from Long-Term Use of Audio-Video Communication*". Computer-Supported Cooperative Work, 5(1):33-62, 1996. ftp://ftp.parc.xerox.com/pub/dourish/jcscw-office-share.ps

[DB92] 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, New York, 1992. http://www.acm.org/pubs/citations/proceedings/chi/142750/p541-dourish/

[DGH+98] T. Darrell, G. Gordon, M. Harville and J. Woodfill. "*Integrated person tracking using stereo, color, and pattern detection*". In Proceedings of the Conference on Computer Vistion and Pattern Recognition, Santa Barbara, pages 601-609, June 1998. http://web.interval.com/papers/1998-021/

[DHT95] C. Diot, C. Huitema, and T. Turletti. "*Multimedia Application should be Adaptive*". In Proceedings of HPCS'95, Mystic (CN), August 1995. ftp://ftp-sop.inria.fr/rodeo/ivs/papers/hpcs95.ps.gz

[Dou93] P. Dourish. "*Culture and Control in a Media Space*". In G. De Michelis, C. Simone, & K. Schmidt, editor, Proceedings of European Conference on Computer-Supported Cooperative Work ECSCW'93, Milano, pages 335-341. Kluwer Academic, September 1993.

ftp://ftp.parc.xerox.com/pub/dourish/ecscw93-culture.ps

[DUF] Dictionnaire Universel Francophone. Hachette/EDICEF/AUPEL-UREF. http://www.francophonie.hachette-livre.fr/

[Egi88] C. Egido. "*Videoconferencing as a Technology to Support Group Work: A Review of its Failure*". Proceedings of the Second Conference on Computer-Supported Cooperative Work (CSCW '88), pages 13-24, September 1988. Held at Portland, Oregon. Published by ACM, New York.. http://www.acm.org/pubs/citations/proceedings/cscw/62266/p13-egido/

[FB96] N. Freed and N. Borenstein. "*Multipurpose Internet Mail Extensions (MIME)*". Draft standard, RFC 2045-2049, IETF, November 1996. http://www.pasteur.fr/infosci/RFC/20xx/

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. "*Hypertext Transfer Protocol - HTTP/1.1*". Draft standard, RFC 2616, IETF, June 1999. http://www.pasteur.fr/infosci/RFC/26xx/2616

[FKC90] R.S. Fish, R.E. Kraut, and B.L. Chalfonte. "*The VideoWindow system in informal communication*". In Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work, pages 1-11. ACM, New York, October 1990. http://www.acm.org/pubs/citations/proceedings/cscw/99332/p1-fish/

[FKR93] R. Fish, R. Kraut, and R. Root. "*Video as a Technology for Informal Communication*". Communications of the ACM, 36(1):48 -60, January 1993. http://www.acm.org/pubs/citations/journals/cacm/1993-36-1/p48-fish/

[FKRR92] R.S. Fish, R.E. Kraut, R.W. Root, and R.E. Rice. "*Evaluating Video as a Technology for Informal Communication*". In Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems, pages 37-48. ACM, New York, 1992. http://www.acm.org/pubs/citations/proceedings/chi/142750/p37-fish/

[FS98] M. Fleischmann and W. Strauss. "*Images of the Body in the House of Illusion*". In C. Sommerer, editor, Art and Science. Springer, 1998. http://imk.gmd.de/images/mars/files/springer_chapter.pdf

[Gas94] J-D. Gascuel. "*Fabule: Un environnement de recherche pour l'animation et la simulation*". In Les Simulateurs, Troisième Séminaire du groupe de travail français Animation/Simulation, Lille, October 1994. http://www-imagis.imag.fr/Publications/jdg/Fabule.Simulateurs94.ps.gz

[Gav92] W.W. Gaver. "*The Affordances of Media Spaces for Collaboration*". In Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work, Toronto, Ontario, pages 17-24. ACM, New York, November 1992. http://www-crd.rca.ac.uk/~bill/refs/vidaff.rtf

[GGR96] S. Greenberg, C. Gutwin, and M. Roseman. "*Semantic Telepointers for Groupware*". In Proc. OzCHI'96, Hamilton, New Zealand, pages 24-27, November 1996. http://www.cpsc.ucalgary.ca/projects/grouplab/papers/1996/96-

SemanticTelepointers.OZCHI/cameraready.pdf

[GKMD94] H. Gajewska, J. Kistler, M.S. Manasse, and D.D.Redell. "*Argo: A System for Distributed Collaboration*". In Proceedings of Multimedia 94, pages 433-440. ACM, New York, October 1994. http://www.acm.org/pubs/citations/proceedings/multimedia/192593/p433-gajewska/

[GMM+92] W.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, New York, 1992. http://www.acm.org/pubs/citations/proceedings/chi/142750/p27-gaver/

[GR80] K. Galloway and S. Rabinowitz. "*Hole-in-space*", 1980. Mobile Image videotape. http://www.ecafe.com/getty/HIS/index.html

[Gre96] S. Greenberg. "*Peepholes: Low Cost Awareness of One's Community*". In Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems, Vancouver, pages 206-207. ACM, New York, April 1996. http://www.cpsc.ucalgary.ca/grouplab/papers/1996/96-ShortPapers.CHI/2-Peepholes/peepholes.pdf

[Gru94] J. Grudin. "*Groupware and social dynamics: Eight challenges for developers*". Communications of the ACM, 37(1):92-105, January 1994. http://www.acm.org/pubs/citations/journals/cacm/1994-37-1/p92-grudin/

[GSHL93] W.W. Gaver, A. Sellen, C. Heath, and P. Luff. "*One is not enough: Multiple views in a Media Space*". In Proceedings of ACM CHI'93 Conference on Human Factors in Computing Systems, Amsterdam, pages 335-341. ACM, New York, April 1993. http://www.acm.org/pubs/citations/proceedings/chi/169059/p335-gaver/

[GSO95] W.W. Gaver, G. Smets, and K. Overbeeke. "*A Virtual Window On Media Space*". In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, Denver, pages 257-264. ACM, New York, May 1995. http://www.acm.org/pubs/citations/proceedings/chi/223904/p257-gaver/

[Gui87] Y. Guiard. "*Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model*". Journal of Motor Behavior, 19:43-61, 1987. http://cogprints.soton.ac.uk/archives/psyc/papers/199803/199803026/doc.html/jmb_87.html

[HCI] "*Curricula for Human-Computer Interaction*". Report of the ACM Special Interest Group on Computer-Human Interaction (SIGCHI) Curriculum Development Group. ACM, New York, 1992. http://www.acm.org/sigchi/cdg/index.html

[HD96] S. Harrison and P. Dourish. "*Re-Place-ing Space: The Roles of Place and Space in Collaborative Environments*". In Proceedings of ACM CSCW'96 Conference on Computer-Supported Cooperative Work, Boston, Mass.. ACM, New York, November 1996. http://www.acm.org/pubs/citations/proceedings/cscw/240080/p67-harrison/

[HPG94] S. Hayne, M. Pendergast, and S. Greenberg. "*Implementing gesturing with cursors in Group Support Systems*". Journal of Management Information Systems, 10(3):43-61, 1994. http://www.cpsc.ucalgary.ca/projects/grouplab/papers/1994/94-Gestures.JMIS/abstract.html

[HS92] 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, New York, 1992. http://www.acm.org/pubs/citations/proceedings/chi/142750/p119-hollan/

[HS96] 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, New York, November 1996. http://www.acm.org/pubs/citations/proceedings/cscw/240080/p248-hudson/

[HSF85] K. Harrenstien, M. Stahl, and E. Feinler. "*NICNAME/WHOIS*". Draft standard, RFC 954, IETF, October 1985. http://www.pasteur.fr/infosci/RFC/9xx/954

[Ish99] 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.

[IKG93] H. Ishii, M. Kobayashi, and J. Grudin. "*Integration of interpersonal space and shared workspace: ClearBoard design and experiments*". ACM Transactions on Information Systems, 11(4):349-375, October 1993. http://www.acm.org/pubs/citations/journals/tois/1993-11-4/p349-ishii/

[IM91] H. Ishii and N. Miyake. "*Toward an open shared workspace: computer and video fusion approach of TeamWorkStation*". Communications of the ACM, 34(12):37-50, December 1991. http://www.acm.org/pubs/citations/journals/cacm/1991-34-12/p37-ishii/

[IT97] E. A. Isaacs and J. Tang. "*Studying video-based collaboration in context: From small workgroups to large organizations*". In K.E. Finn, A.J. Sellen, and S.B. Wilbur, editors, Video-mediated communication, pages 173-197. Lawrence Erlbaum Associates, 1997. http://www.izix.com/pro/videouse.html

[IWFO97] E. A. Isaacs, S. Whittaker, D. Frohlich, and B. O'Conaill. "*Informal communication re-examined: New functions for video in supporting opportunistic encounters*". In K.E. Finn, A.J. Sellen, and S.B. Wilbur, editors, Video-mediated communication. Lawrence Erlbaum Associates, 1997. http://earthlight.com/ellen/papers/oppcomm-vmc.html

[JG99] 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, January 1999. http://www.cpsc.ucalgary.ca/grouplab/papers/1999/99-JudgingAvailability.HICSS/camera-ready-hicss.pdf

[Joh85] M. St Johns. "*Authentication Server*". Technical Report RFC 931, IETF, January 1985.

http://www.pasteur.fr/infosci/RFC/9xx/931

[Kar94] A. Karsenty. "*Le collecticiel : de l'interaction homme-machine à la communication homme-homme*". Technique et Science Informatique. 13(1) : 105-127, Hermès, 1994.

[KL86] B. Kantor and P. Lapsley. "*Network News Transfer Protocol*". Proposed standard, RFC 977, IETF, February 1986. http://www.pasteur.fr/infosci/RFC/9xx/977

[KM97] D. Kristol and L. Montulli. "*HTTP State Management Mechanism*". Proposed Standard, RFC 2109, IETF, February 1997. http://www.pasteur.fr/infosci/RFC/21xx/2109

[Kru91] M. Krueger. Artificial Reality II. Addison-Wesley, 1991.

[KS97] E. Kandogan and B. Shneiderman. "*Elastic Windows: evaluation of multi-window operations*". In Proceedings of the CHI'97 conference on Human Factors in computing systems, Atlanta, pages 250-257. ACM, New York, March 1997. http://www.acm.org/pubs/citations/proceedings/chi/258549/p250-kandogan/

[Kub68] S. Kubrick. 2001: A Space Odyssey, 1968. Metro-Goldwyn-Mayer.

[LGS97] A. Lee, A. Girgensohn, and K. Schlueter. "*NYNEX Portholes: Initial User Reactions and Redesign Implications*". In Proceedings of GROUP'97, Phoenix, pages 385-394. ACM, 1997. http://www.acm.org/pubs/citations/proceedings/cscw/266838/p385-lee/

[Lyl93] B. Lyles. "*Media Spaces and Broadband ISDN* ". Communications of the ACM, 36(1):46-47, January 1993. http://www.acm.org/pubs/citations/journals/cacm/1993-36-1/p28-bly/

[LZB98] A. Leganchuk, S. Zhai and W. Buxton. "*Manual and cognitive benefits of two-handed input: an experimental study*". ACM Transactions on Computer-Human Interaction, 5(4):326-359, March 1998. http://www.acm.org/pubs/articles/journals/tochi/1998-5-4/p326-leganchuk/p326-leganchuk.pdf

[Mac88] W. Mackay. "*More than Just a Communication System: Diversity in the Use of Electronic Mail*". In Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work. ACM, New York, September 1988. http://www.acm.org/pubs/citations/proceedings/cscw/62266/p344-mackay/

[Mac90] W. Mackay. "*Users and Customizable Software: A Co-Adaptive Phenomenon*", PhD thesis, Massachusetts Instititute of Technology (USA), 1990.

[Mac96] W. Mackay. "*Réalité augmentée : le meilleur des deux mondes*". La Recherche, numéro spécial "L'ordinateur au doigt et à l'oeil", pages 32-37, mars 1996.

[Mac99] 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.

[Mas98] L. Masinter. "*The data URL scheme*". Proposed standard, RFC 2397, IETF, August 1998.
http://www.pasteur.fr/infosci/RFC/23xx/2397

[MBK+97] S. McCanne, E. Brewer, R. Katz, L. Rowe, E. Amir, Y. Chawathe, A. Coopersmith, K. Mayer-Patel, S. Raman, A. Schuett, D. Simpson, A. Swan, T-L. Tung, D. Wu, and B. Smith. "*Toward a Common Infrastructure for Multimedia-Networking Middleware*". In Proceedings of 7th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 97), may 1997.
http://www-mash.cs.berkeley.edu/dist/mash/papers/mash-nossdav97.ps.gz

[MBS+91] M.M. Mantei, R.M. Baecker, A.J. Sellen, W.A.S. Buxton, and T. Milligan. "*Experiences in the Use of a Media Space*". In Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, pages 203-208. ACM, New York, 1991.
http://www.acm.org/pubs/citations/proceedings/chi/108844/p203-mantei/

[MCLM90] A. Maclean, K. Carter, L. Lövstrand, and T. Moran. "*User tailorable systems: Pressing the issues with Buttons*". In Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems, Seattle, pages 175-182. ACM, New York, April 1990.
http://www.acm.org/pubs/citations/proceedings/chi/97243/p175-maclean/

[MJ95] S. McCanne and V. Jacobson. "*Vic: A Flexible Framework Framework for Packet Video*". In Proceedings of the third international conference on Multimedia '95, pages 511-522. ACM, New York, 1995.
http://www.acm.org/pubs/citations/proceedings/multimedia/217279/p511-mccanne/

[MK93] M. J.Muller and S. Kuhn. "*Participatory design*". Communications of the ACM, 36(6):24-28, June 1993.
http://www.acm.org/pubs/citations/journals/cacm/1993-36-6/p24-muller/

[MS94] G. Moore and K. Schuyler. "*Videoconferencing 1990s Style: Sharing faces, Places and Spaces*". Technical Report OTP-94-03, Ontario Telepresence Project, Toronto, 1994.
http://www.dgp.utoronto.ca/tp/papers/9403.html

[MVC+93] W. Mackay, G. Velay, K. Carter, C. Ma, and D. Pagani. "*Augmenting Reality: Adding Computational Dimensions to Paper*". Communications of the ACM, 36(7):96-97, July 1993.
http://www.acm.org/pubs/citations/journals/cacm/1993-36-7/p96-mackay/

[Net94] J. Zawinski. "*Remote control of UNIX Netscape*". Technical report, Netscape Communications, December 1994.
http://home.netscape.com/newsref/std/x-remote.html

[Net95] "*An Exploration of Dynamic Documents*". Technical report, Netscape Communications, 1995.
http://home.netscape.com/assist/net_sites/pushpull.html

[Nor94] D.A. Norman. "*Things That Make Us Smart : Defending Human Attributes in the Age of the Machine*". Perseus Pr, May 1994.

[Nor96] D.A. Norman. "*Grandeur et misère de la technologie. Les relations entre l'homme et la machine ne sont pas idéales. A qui la faute ?* ". La Recherche, numéro spécial "L'ordinateur au doigt et à l'oeil", pages 22-25, mars 1996.

[ND86] D.A. Norman and S.W. Drapper, editors. "*User Centered System Design: New Perspectives on Human-Computer Interaction*". Hillsdale, NJ: Lawrence Erlbaum Associates.

[Nou98] G. Nouvel. "*Pointage à distance avec la main par la technique du Chroma Keying*". Rapport de stage de DEA, Université Paris-Sud, France, Septembre 1998.

[Nør91] T. Nørretranders. The user illusion: cutting consciousness down to size. Penguin books, 1991.

[OB91] M. Olson and S. Bly. "*The Portland Experience: a report on a distributed research group*". International Journal of Man-Machine Studies, 34:211-228, 1991. Published by Academic Press Limited.

[PG92] P. Perin and M. Gensollen, éditeurs. "*La communication plurielle: l'interaction dans les téléconférences*". La documentation Française , 1992.

[Pos80] J. Postel. "*User Datagram Protocol*". Standard, RFC 768, IETF, August 1980. http://www.pasteur.fr/infosci/RFC/7xx/768

[Pos81a] J. Postel. "*Internet Protocol*". Standard, RFC 791, IETF, September 1981. http://www.pasteur.fr/infosci/RFC/7xx/791

[Pos81b] J. Postel. "*Transmission Control Protocol*". Standard, RFC 793, IETF, September 1981. http://www.pasteur.fr/infosci/RFC/7xx/793

[Pos82] J. Postel. "*Simple Mail Transfer Protocol*". Standard, RFC 821, IETF, August 1982. http://www.pasteur.fr/infosci/RFC/8xx/821

[Pri99] W. Prinz. "*NESSIE: An Awareness Environment for Cooperative Settings*", in Proceedings of The Sixth European Conference on Computer Supported Cooperative Work - ECSCW'99, S. Bødker, M. Kyng, and K. Schmidt (eds.), Kluwer Academic Publishers, 1999, pp 391-410. http://orgwis.gmd.de/~prinz/pub/ECSCW99-Nessie.pdf

[PR83] J. Postel and J. Reynolds. "*Telnet Protocol*". Standard, RFC 854, IETF, May 1983. http://www.pasteur.fr/infosci/RFC/8xx/854

[PR85] J. Postel and J. Reynolds. "*File Transfer Protocol (FTP)* ". Standard, RFC 959, IETF, October 1985. http://www.pasteur.fr/infosci/RFC/9xx/959

[PS97] E. Rønby Pedersen and T. Sokoler. "*AROMA: abstract representation of presence supporting mutual awareness*". In Proceedings of the CHI'97 conference on Human Factors in computing systems, Atlanta, pages 51-58. ACM, New York, March 1997. http://www.acm.org/pubs/citations/proceedings/chi/258549/p51-pedersen/

[RG93] M. Roseman and S. Greenberg. "*Building Flexible Groupware Through Open Protocols*". In ACM Conference on Organizational Computing Systems,

California, pages 279-288. ACM Press, October 1993.
http://www.acm.org/pubs/citations/proceedings/cocs/168555/p279-roseman/

[RG96] M. Roseman and S. Greenberg. "*Building Real Time Groupware with GroupKit, A Groupware Toolkit*". ACM Transactions on Computer-Human Interaction, 3(1):66-106, March 1996.
http://www.acm.org/pubs/citations/journals/tochi/1996-3-1/p66-roseman/

[RHJ99] D. Raggett, A. Le Hors, and I. Jacobs. "*HTML 4.01 Specification*". Technical report, W3C User Interface Domain Recommendation, December 1999.
http://www.w3.org/TR/html4/. http://www.w3.org/TR/html4/

[Rie96] R. Riesenbach. "*Less is more (more or less...)* ". Telepresence Systems White Paper, 1996.
http://www.videoconference.com/lessis.htm

[Roc98] E. Rocco. "*Trust breaks down in electronic contexts but can be repaired by some initial face-to-face contact*". In Proceedings of ACM CHI'98 Conference on Human Factors in Computing Systems, Los Angeles, pages 496-502. ACM, New York, April 1998.
http://www.acm.org/pubs/citations/proceedings/chi/274644/p496-rocco/

[RN99] N. Roussel and G. Nouvel. "*La main comme télépointeur*". In Tome 2 des actes des onzièmes journées francophones sur l'Interaction Homme Machine (IHM'99), Montpellier , Novembre 1999.

[Rou99a] N. Roussel. "*Beyond Webcams and Videoconferencing: Informal Video Communication on the Web*". In British HCI Conference on The Active Web, Stafford , January 1999.

[Rou99b] N. Roussel. "*Mediascape: a Web-based Mediaspace*". IEEE Multimedia, 6(2), April-June 1999.
http://www.computer.org/multimedia/mu1999/u2064abs.htm

[RSFWH98] T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. "*Virtual Network Computing*". IEEE Internet Computing, 2(1):33-38, Jan-Feb 1998.
ftp://ftp.uk.research.att.com/pub/docs/att/tr.98.1.pdf

[RvDR+00] 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 the CHI 2000 conference on Human Factors in computing systems, pages 494-501. ACM, New York, April 2000.
http://www.acm.org/pubs/citations/proceedings/chi/332040/p494-robertson/

[SC93] D. Salbert and J. Coutaz. "*Groupe+Video+Audio+Informatique=Mediaspace. Etat de l'art et revue de problèmes*". In Rapport de l'atelier Collecticiel des journées IHM'93, Lyon , pages 67-72, 1993.
http://iihm.imag.fr/publs/1993/IHM93_MediaSpace.Fr.ps.gz

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "*RTP: A Transport Protocol for Real-Time Applications*". Proposed Standard, RFC 1889, IETF, January 1996.
http://www.pasteur.fr/infosci/RFC/18xx/1889

[SGH+99] N.A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. "*i-LAND: An interactive Landscape for Creativitiy and Innovation*". In Proceedings of ACM CHI'99 Conference on Human Factors in Computing Systems, Pittsburgh, pages 120-127. ACM, New York, May 1999. http://www.darmstadt.gmd.de/ambiente/paper/chi99Reprint.pdf

[Sti97] O. Stiemerling. "*Supporting Tailorability in Groupware through Component Architectures*". In Proceedings of the ECSCW '97 Workshop on Object Oriented Groupware Platforms, Lancaster, GB, pages 53-57, September 1997. http://www.informatik.uni-bonn.de/~os/Publications/OOGP97.ps

[Stu86] R. Stults. "*Media Space*". Technical report, Xerox PARC, 1986.

[Suc83] L. Suchman. "*Office procedures as practical action: Models of work and system design*". ACM Transactions on Office Information Systems, 1:320-328, 1983.

[SP96] C. Szyperski and C. Pfister. "*Component-Oriented Programming: WCOP '96 Workshop Report*". Special Issues in Object-Oriented Programming, Workshop Reader of the 10th European Conference on Object-Oriented Programming ECOOP '96, Linz, pp. 127-130, 1996.

[TB94] T. Trowt-Bayard. Videoconferencing: The Whole Picture. Flatiron Publishing, 1994.

[TI93] J.C. Tang and E.A. Isaacs. "*Why Do Users Like Video ? Studies of Multimedia-Supported Collaboration*". In Computer Supported Cooperative Work: An International Journal, pages 163-196. Vol. 1, Issue 3, 1993. http://www.sunlabs.com/technical-reports/1992/abstract-5.html

[TM91] J.C. Tang and S.L. Minneman. "*VideoDraw: A Video Interface for Collaborative Drawing*". ACM Transactions on Information Systems, 9(2):170-184, April 1991. http://www.acm.org/pubs/citations/journals/tois/1991-9-2/p170-tang/

[TR94] J.C. Tang and 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, New York, April 1994. http://www.acm.org/pubs/citations/proceedings/chi/191666/p37-tang/

[TR00] M. Turk and G. Robertson. "*Perceptual user interfaces (introduction)*". Communications of the ACM, 43(3):32-34, March 2000. http://www.acm.org/pubs/citations/journals/cacm/2000-43-3/p32-turk/

[Tur95] T. Turletti. "*Controle de Transmission pour Logiciel de Videoconference*". Thèse de nouveau doctorat, Université de Nice-Sophia Antipolis (France), Avril 1995.

[Veg96] A. Vega-Garcia. "*Mécanismes de Contrôle pour la Transmission de l'Audio sur l'Internet*". Thèse de nouveau doctorat, Université de Nice-Sophia Antipolis (France), Octobre 1996.

[VLNC99] F. Vernier, C. Lachenal, L. Nigay, and J. Coutaz. "*Interface augmentée par effet miroir*". In Actes des onzièmes journées francophones sur l'Interaction Homme Machine (IHM'99), Montpellier, pages 158-165, Novembre 1999.

http://iihm.imag.fr/publs/1999/IHM99_Interfaces_Augmentees.pdf

[Weg97] P. Wegner. "*Why interaction is more powerful than algorithms*". Communications of the ACM, 40(5):80-91, May 1997. http://www.acm.org/pubs/citations/journals/cacm/1997-40-5/p80-wegner/

[Wei91] M. Weiser. "*The Computer for the Twenty-First Century*". Scientific American, 265(3):94-104, September 1991. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Wel93] P. Wellner. "*Interacting with paper on the Digital Desk*". Communications of the ACM, 36(7):87-96, July 1993. http://www.acm.org/pubs/citations/journals/cacm/1993-36-7/p87-wellner/

[WSM+90] K. Watabe, S. Sakata, K. Maeno, H. Fukuoka, and T. Ohmori. "*Distributed multiparty desktop conferencing system: MERMAID*". In Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work, pages 27-38. ACM, New York, October 1990. http://www.acm.org/pubs/citations/proceedings/cscw/99332/p27-watabe/

[Zim91] D. Zimmerman. "*The Finger User Information Protocol*". Draft standard, RFC 1288, IETF, December 1991. http://www.pasteur.fr/infosci/RFC/12xx/1288

Remarque : la numérotation des pages de ce document est différente de celle de la version «officielle» de la thèse tirée par l'Université Paris-Sud.