



Interactions post-WIMP et applications existantes sur une table interactive

Guillaume Besacier

► To cite this version:

Guillaume Besacier. Interactions post-WIMP et applications existantes sur une table interactive. Informatique [cs]. Université Paris Sud - Paris XI, 2010. Français. NNT : . tel-00596508

HAL Id: tel-00596508

<https://theses.hal.science/tel-00596508>

Submitted on 27 May 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-SUD 11

Discipline : Informatique

préparée au *Laboratoire de recherche en informatique*
et au *Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur*

dans le cadre de l'École doctorale d'informatique de Paris-Sud (ED427).

présentée et soutenue publiquement par

GUILLAUME BESACIER

le 19 octobre 2010

Titre :

**Interactions post-WIMP
et applications existantes
sur une table interactive**

Directeurs de thèse :

Michel Beaudouin-Lafon
Frédéric Vernier

Jury :

Michel Beaudouin-Lafon	Directeur de thèse
François Bérard	Examineur
Patrick Girard	Rapporteur
Franck Poirier	Rapporteur
Frédéric Vernier	Directeur scientifique
Anne Vilnat	Présidente du jury

Remerciements

Je tiens à remercier chaleureusement mes directeurs de thèse, Michel et Frédéric. Il ne le sait pas, mais c'est une démo de Frédéric alors qu'il était chargé de TD à l'école d'ingénieur où j'étais étudiant qui est la principale raison qui a motivé l'existence de cette thèse. Je veux être ingénieur depuis longtemps. On pourrait dire que ça coule dans ma famille : mon oncle est ingénieur spécialiste en bâtiments parasismiques, il vient de passer plusieurs années au Sri Lanka à étudier les effets du tsunami de 2004 ; mon grand-père maternel était ingénieur à la ville de Paris, où il a gravi les échelons de assistant-géomètre à directeur de l'assainissement ; mon grand-père paternel, que j'ai peu connu, était ingénieur-slash-inventeur-slash-entrepreneur, auteur de plusieurs brevets a priori toujours utilisés aujourd'hui, même je me souviens surtout de ses petits poissons en plastique qui nageaient tout seuls. J'en ai démonté des dizaines quand j'avais 7 ou 8 ans, sans jamais réussir à comprendre comme ils s'animaient sans moteur. Mon idée de l'ingénierie s'était formée autour d'ingénieurs comme Gustave Eiffel, qui ont repoussé les limites de la connaissance en appliquant les technologies les plus innovantes du moment à des problèmes inédits et grandioses. Aussi, quand après une formation en informatique de tout premier plan, l'école d'ingénieur nous a annoncé que nous commençons les matières « sérieuses », qualité ISO neuf-mille-et-des-poussières, méthode COCOMO pour devinestimer le coût d'un projet (je maintiens que l'on peut obtenir des résultats tout aussi probants en jetant le cahier des charges dans l'escalier et en comptant les marches), le tout saupoudré de comptabilité analytique et de droit du travail, j'ai commencé à comprendre que nous étions plus formés à devenir des PHB que des Dilbert. Le coup de grâce, si je puis dire, à mes idées d'ingénierie innovante est ma rencontre, lors d'un stage, avec un manager qui faisait des bases de données avec Excel et beaucoup d'huile de macro VBA. J'ai bien essayé de le convaincre de passer à Access (Access ! Pas Oracle, Access), mais il trouvait ça trop difficile. J'ai appris plus tard que ce manager était en ancien élève de l'école d'ingénieur où j'étais. Aussi, quand Frédéric nous a présenté, un jour de TD de Java-Blobby-man, son tableur avec un fish-eye, son moteur de recherche en 3D, ses impayables grimaces, et surtout son projet de table interactive, avec un enthousiasme communicatif, j'ai su que c'est dans la recherche publique qu'il fallait que je sois. Je n'ai pas encore mentionné que la moitié de ma famille qui n'est pas dans l'ingénierie est dans l'enseignement. On ne pouvait rêver plus « reprise du flambeau » que l'enseignement supérieur et la recherche.

Je tiens à remercier ma famille, justement, qui m'a soutenu durant ces quatre années. Merci en particulier à ma mère, qui a supporté de relire ce manuscrit parfois un peu technique, et a offert des commentaires très pertinents sur le degré de (non-)lisibilité de certains passages.

Je remercie les rapporteurs de cette thèse, Patrick Girard et Franck Poirier, pour leurs commentaires, toujours intéressants et plein d'à propos. Je les remercie aussi d'avoir accepté cette tâche.

Je remercie les membres de mon jury, Michel Beaudouin-Lafon, François Bérard, Patrick Girard, Franck Poirier, Frédéric Vernier, et sa présidente Anne Vilnat, pour avoir lu attentivement mes travaux, pour s'être déplacés parfois de loin pour m'écouter et pour m'avoir fait part de remarques constructives et de leurs questions à la difficulté... bien dosée.

Je remercie enfin tous les membres d'AMI et d'in-situ, en particulier ceux avec qui j'ai travaillé, Gaëtan, Jonathan, Nicolas, Olivier et Rami, ainsi que Stéphanie, Marianne, Asimina et Pierre de l'ENSAM.

Ceux que j'ai oubliés se reconnaîtront.

Nos travaux s'inscrivent dans la discipline de l'interaction homme-machine (IHM), et concernent l'utilisation d'une nouvelle « forme » de systèmes informatiques : les tables interactives ou « tabletop ». Les systèmes tabletop prennent la forme d'une table dans son entier, qui sert à la fois de périphérique d'entrée tactile et de périphérique de sortie graphique. Grâce aux technologies multi-tactile, plusieurs utilisateurs peuvent toucher la table en même temps, et interagir avec le système de façon indépendante. Cette propriété rend les tables interactives particulièrement adaptées au support de la collaboration. Pourtant, nous constatons que les tables interactives ne sont que peu utilisées dans un objectif collaboratif en dehors des laboratoires. Nous pensons que plusieurs causes permettent d'expliquer cet état de fait.

Nous remarquons, d'une part, que les applications existantes, pour ordinateurs de bureau, forment une part importante de l'utilisation actuelle des systèmes informatiques. Ces applications, qu'elles soient des applications grand public largement utilisées ou des applications spécifiques à une tâche précise, sont indispensables à leurs utilisateurs. Des projets ont créés des nouvelles applications aux mêmes fonctionnalités que les applications existantes, mais adaptées à des prototypes de nouveaux systèmes d'interaction. Elles offrent nettement moins de fonctionnalités que les applications pour ordinateurs de bureau existantes. Une telle réécriture reste de plus coûteuse au regard des efforts de développement déjà investis dans la création des applications existantes.

D'autre part, les tables interactives se présentent sous un aspect matériel différent des ordinateurs traditionnels, au-delà du trio écran-clavier-souris. Cela introduit de nouvelles possibilités mais aussi de nouveaux défis pour l'interface utilisateur des applications. En revanche, les besoins des utilisateurs, en termes de fonctionnalités des applications, restent les mêmes. Nous étudions comment ce changement peut se répercuter sur les applications existantes. Nous proposons d'utiliser les mêmes applications sur la table interactive que sur l'ordinateur de bureau. Plusieurs types d'interactions sont envisageables pour remplacer les styles d'interaction de l'ordinateur de bureau qui ne sont pas adaptés aux systèmes tabletops. Nous entrons ici dans le domaine des interactions post-WIMP, des interactions qui vont au-delà des interacteurs 2D classiques que sont les fenêtres, les icônes, les menus et le pointeur (WIMP : Windows, Icons, Menus, Pointer).

Les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau. Nous évaluons six technologies permettant faire le lien entre les techniques utilisées pour programmer des applications pour ordinateurs de bureau et les techniques utilisées pour programmer des applications pour tables interactives : capture d'écran, carte graphique virtuelle, clavier virtuel et souris virtuelle, langage de script, API d'accessibilité numérique, réécriture d'une boîte à outils d'interface homme machine. Pour les évaluer, nous définissons cinq axes (structuration des données, flexibilité, performance,

compatibilité et réutilisabilité, et difficulté d'implémentation). Nous avons implémenté des prototypes pour la carte graphique virtuelle, le langage de script, et la réécriture d'une boîte à outils (la boîte à outils win32).

Les applications pour tables interactives ne s'utilisent pas de la même façon que les applications pour ordinateurs de bureau non plus. Les interfaces utilisateur des applications existantes ont été conçues pour être utilisées par un seul utilisateur avec un ordinateur de bureau. Les technologies que nous avons présentées ne résolvent pas ce problème. Nous présentons des scénarios mettant en évidence ces limitations.

Pour concevoir de nouvelles techniques d'interaction adaptées, nous introduisons la métaphore du papier. Les métaphores sont utilisées, en interaction homme-machine, pour aider les utilisateurs à appréhender le fonctionnement d'un nouveau système plus rapidement en y appliquant des connaissances d'un système qu'ils connaissent déjà. Le papier, probablement le support d'information le plus courant, offre de nombreuses possibilités d'interaction. Nous proposons ainsi trois techniques d'interaction inspirées du papier. Nous proposons des fentes qui permettent de transmettre à distance des documents à un autre utilisateur, ainsi que d'effectuer des traitements sur les documents. Le pliage durable permet de nouvelles interactions avec un document. Il rend accessible le dos du document, qui peut contenir une interface graphique pour l'utilisateur. En observant le recto de documents papier existants, nous sommes arrivés à la conclusion que certains artefacts informatiques existants auraient toute leur place au dos d'un document : le système d'aide de l'application, des outils pour gérer le cycle de vie du document, des métadonnées sur le document,... Enfin, le plier pour empiler permet la création de piles de façon spontanée, en « recyclant » un document existant comme couverture pour de pile, simplement en le pliant en deux.

Certaines des interactions peuvent être effectuées de plusieurs manières. Nous proposons un formalisme que nous appelons matrice des interactions, dédié aux interactions de gestion des documents sur la surface d'une table interactive, pour les comparer. Nous nous basons sur les interactions déjà existantes dans la gestion des fenêtres sur les ordinateurs de bureau : le déplacement et le redimensionnement, auxquelles nous ajoutons trois interactions : la rotation, le zoom et le pliage. Ces interactions s'approchent de devenir des standards en matière d'interface pour table interactive dans la communauté des *tabletops*. Nous avons identifié trois moyens distincts d'effectuer ces cinq interactions : l'utilisation d'un interacteur dédié, l'utilisation de la position du document sur la table, et l'utilisation de la trajectoire du doigt de l'utilisateur. Notre matrice met en évidence que ces trois moyens peuvent être proposés simultanément par le système informatique, imitant ainsi cette propriété du monde physique.

Introduction	1
Constat.....	3
Contributions	6
Plan de ce mémoire	7
 État de l'art	 9
 Chapitre 1 : Amener les applications existantes sur les tables interactives	 19
1.1. Introduction	21
1.2. Axes	23
1.3. Technologies	26
1.3.1. Capture d'écran	26
1.3.2. Carte graphique ou écran virtuels	26
1.3.3. Clavier virtuel et une souris virtuelle	29
1.3.4. Langage de script.....	30
1.3.5. Accessibilité numérique.....	32
1.3.6. Modification d'une boîte à outils d'interface homme machine	34
1.3.6.1. Choix d'une boîte à outils.....	38
1.3.6.2. Lien entre la nouvelle boîte à outils et les applications existantes.....	38
1.3.6.3. Types de modifications.....	40
1.3.6.4. Implémentation.....	44
1.4. Conclusion	50
 Chapitre 2 : Interactions sur les tables interactives et logiciels existants.....	 53
2.1. Introduction.....	55
2.2. Scénarios	57
2.2.1. Édition d'images	57
2.2.2. Édition d'une feuille de calcul.....	59
2.3. Conclusion	63

Chapitre 3 : Gestion de l'espace sur les tables interactives	65
3.1. Introduction.....	67
3.2. Métaphore du papier.....	70
3.3. Fentes	75
3.4. Pliage	78
3.4.1. Activation du pliage durable.....	81
3.4.2. Interaction avec un document plié	88
3.5. Plier pour empiler	98
3.5.1. Activation du plier pour empiler	102
3.5.2. Aspects graphiques de la métaphore	105
3.5.3. Implémentation.....	106
3.6. Conclusion	111
 Chapitre 4 : Différents moyens d'activer différentes interactions : la matrice des interactions	 113
4.1. Introduction.....	115
4.2. Interacteur dédié	117
4.3. Position du document sur la table	119
4.4. Trajectoire du doigt.....	124
4.5. Conclusion	128
 Conclusion.....	 131
Contributions	133
Perspectives	136
A plus long terme	138
 Bibliographie.....	 139
 Index des figures	 147

Introduction_____

Nos travaux s'inscrivent dans la discipline de l'interaction homme-machine (IHM), et concernent l'utilisation d'une nouvelle « forme » de systèmes informatiques : les tables interactives ou « *tabletop* ». Les systèmes *tabletop* sont nommés ainsi par analogie avec les ordinateurs *desktop* (de bureau), *laptop* (portable, littéralement « de genoux »), et *palmtop* (littéralement « dans la main »). Les systèmes *tabletop* prennent la forme d'une table dans son entier, qui sert à la fois de périphérique d'entrée tactile et de périphérique de sortie graphique. Autrement dit, l'équivalent de l'écran, du clavier et de la souris des ordinateurs de bureau se trouvent combinés en un périphérique unique : la table.

Grâce aux technologies multi-tactile, plusieurs utilisateurs peuvent toucher la table en même temps. Dans ce schéma d'utilisation, chacun peut interagir avec le système de façon indépendante des autres utilisateurs. La même surface horizontale peut donc être partagée par plusieurs utilisateurs du système, et chaque utilisateur peut utiliser une ou ses deux mains et/ou un ou plusieurs doigts pour interagir avec le système. Les schémas d'utilisation sont nombreux et couvrent, par exemple, les gestes bi-manuel. Cette propriété rend les tables interactives particulièrement adaptées au support de la collaboration au sein d'un groupe dans le but d'accomplir une tâche commune. Nous remarquons également que l'utilisation du contact direct (sans souris, ni pointeur) apporte un gain d'intuitivité et un moyen pour les utilisateurs de facilement faire partager leurs connaissances du système. Par exemple, un utilisateur peut montrer à un autre comment utiliser le système, et ce dernier apprendre par imitation, ce qui n'est pas facile dans une configuration avec une souris et un clavier. Pourtant, nous constatons que les tables interactives ne sont que peu utilisées dans un objectif collaboratif en dehors des laboratoires. Nous pensons que plusieurs causes permettent d'expliquer cet état de fait.

Nous remarquons, d'une part, que les applications existantes, pour ordinateurs de bureau, forment une part importante de l'utilisation actuelle des systèmes informatiques, tant dans le domaine de la vie professionnelle que des loisirs. Ces applications, qu'elles soient des applications grand public largement utilisées ou des applications spécifiques à une tâche précise, sont indispensables à leurs utilisateurs. Nous observons que ces applications font aujourd'hui partie d'un ensemble d'applications interagissant autour de flux de données, dont de larges quantités de données existantes. Nous traduisons ce besoin d'interopérabilité par la nécessité d'un système qui soit utile et utilisable. Si les tables interactives sont intéressantes pour la collaboration, il est tout aussi important que les tables interactives permettent aux utilisateurs d'accéder aux outils qui rendent le système utile. Il est envisageable de réécrire, au moins partiellement, les applications les plus utilisées, telles que les traitements de texte ou les navigateurs internet, pour les rendre compatibles avec un environnement de tables interactives. Certains projets ont créé de nouvelles applications aux mêmes fonctionnalités que les applications existantes, mais adaptées à des prototypes de nouveaux systèmes d'interaction, dont les tables interactives font partie. Les concepteurs de la boîte à outils pour tables interactives

T3 [120] ont confié à des étudiants la tâche d'écrire un tableur, un navigateur web, et plusieurs autres applications utilisant T3. Des applications telles que *DTMap* [39], ou les innombrables applications de partage d'images pour tables interactives offrent nettement moins de fonctionnalités que les applications pour ordinateurs de bureau existantes. Une telle réécriture reste coûteuse au regard des efforts de développement déjà investis dans la création des applications existantes et les grandes entreprises de l'informatique qui développent et distribuent ces applications sont très réticentes à donner accès aux informations nécessaires pour écrire une application compatible. En revanche, les applications spécialisées sont trop nombreuses pour être toutes réécrites et recrées. Nous constatons déjà que de nombreuses entreprises repoussent la migration vers de nouvelles versions d'un même système d'exploitation lorsque cette migration oblige à remettre en question des applications dédiées de l'entreprise. Les entreprises ne peuvent pas adopter de façon massive une nouvelle forme de systèmes informatiques, comme les *tabletops*, si ceux-ci ne s'interfacent pas avec leurs processus, leurs applications et leurs données existantes de flux de travail, de gestion de biens, de comptabilité, de modélisation ou encore de productivité. Ces applications incluent des applications spécifiques à une entreprise, ou développées en interne, qui n'en sont pas moins indispensables pour leurs utilisateurs. Il serait donc déraisonnable de songer à réécrire les applications existantes pour les faire fonctionner dans l'environnement offert par un système *tabletop*. Seuls les professionnels concernés peuvent de plus adapter leurs propres applications, or nous nous autorisons à penser qu'ils ne le feront pas : en effet, si certaines organisations ne mettent pas à jour leurs systèmes informatiques de bureau vers les nouvelles versions de ces mêmes systèmes parce que leurs applications pourraient ne pas le supporter, il est illusoire de penser qu'elles seraient prêtes à réécrire ces applications, avec le risque d'introduire de nouveaux défauts, pour un nouveau type de système.

D'autre part, les tables interactives se présentent sous un aspect matériel différent des ordinateurs traditionnels (de bureau ou portable), au-delà du trio écran-clavier-souris. Cela introduit de nouvelles possibilités mais aussi de nouveaux défis pour l'interface utilisateur des applications [125]. En revanche, les besoins des utilisateurs, en termes de fonctionnalités des applications, restent les mêmes. Nous avons choisi dans ce mémoire d'étudier comment ce changement peut se répercuter sur les applications existantes. Prenons par exemple les travaux sur les interacteurs tangibles et en particulier sur l'utilisation d'un interacteur tangible pour représenter un dossier [121]. Ces travaux décrivent comment l'utilisateur pose un interacteur sur un document numérique présent sur une table pour activer la fonction « sauvegarder ce document dans ce dossier ». Cette fonctionnalité est identique à l'utilisation d'une souris pour cliquer sur un bouton « sauvegarder » dans une barre d'outils et de sélectionner un dossier avec une boîte de dialogue. Si les fonctionnalités restent les mêmes, nous pouvons envisager d'utiliser les mêmes applications sur la table interactive que sur l'ordinateur de bureau. Plusieurs types d'interactions sont toutefois envisageables pour remplacer les styles d'interaction de l'ordinateur de bureau qui ne sont pas adaptés aux systèmes *tabletops*. Nous entrons ici dans le domaine des interactions post-WIMP [122], des interactions qui vont au-delà des

interacteurs 2D classiques que sont les fenêtres, les icônes, les menus et le pointeur (WIMP : *Windows, Icons, Menus, Pointer*).

En plus de permettre de répondre aux besoins des utilisateurs de tables interactives, la possibilité d'utiliser des applications existantes avec un système *tabletop* ouvrirait la porte aux études in situ à long terme. En effet, la pratique d'études sur le terrain et sur une période significative de temps fait partie de l'arsenal scientifique depuis la fin du XVIII^e siècle [67]. Elle est indispensable pour la compréhension de l'interaction homme machine. Elle a par exemple été utilisée avec succès pour valider des lois empiriques [16] ou étudier les usages de l'utilisateur [51]. Ces deux études ont duré plusieurs mois. Dans le domaine des tables interactives, à cause du manque d'interopérabilité avec les applications et les données existantes, la seule étude de terrain à notre connaissance [125] n'a porté que sur un seul utilisateur, utilisant une table interactive presque comme un ordinateur de bureau. La possibilité d'utiliser les applications supportant le travail des utilisateurs dans des conditions d'interaction adaptées aux tables interactives permettrait par exemple d'étudier l'utilisation multi-utilisateur et tous les aspects sociaux que cela implique.

Contributions

Nous proposons dans cette thèse d'étudier les différentes possibilités d'insérer une couche logicielle d'adaptation entre les applications existantes et l'ordinateur *tabletop*. En effet, l'utilisation d'applications existantes sur les tables interactives commence simplement avec le problème de pouvoir faire utiliser l'environnement système de la table à l'application. Les tables interactives ne sont pas des ordinateurs de bureau, elles ont des fonctionnalités différentes, et donc les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau. Nous définissons plusieurs métriques pour comparer les différentes façons d'accomplir cette interopérabilité. Nos métriques mesurent la structuration des données, la flexibilité, la performance, la compatibilité et réutilisabilité, et la difficulté d'implémentation. Nous les appliquons à plusieurs stratégies, qui sont autant de technologies logicielles permettant de créer un lien entre une application existante et l'environnement système d'une table interactive.

D'autre part, l'interface utilisateur des applications existantes, et les boîtes à outils d'interaction homme-machine qu'elles utilisent, ne sont pas toujours adaptés à leur utilisation sur une surface horizontale, grande et partagée par plusieurs utilisateurs. Nous nous proposons de catégoriser les opérations de gestion de fenêtres d'un côté, et les types d'interaction adaptées aux tables interactives de l'autre pour générer une relation entre les deux qui permette de substituer dynamiquement plusieurs types d'interactions post-WIMP aux interactions WIMP de l'ordinateur de bureau. En présentant ces possibilités sous la forme d'une matrice qui reprend pour chaque opération identifiée son équivalent en suivant l'inspiration ou la métaphore qui domine chaque type d'interactions post-WIMP, nous permettons à la fois de rendre cette substitution automatisable, ceci sans modifier les applications existantes, et de générer de nouvelles interactions dans les « cases vides » de la matrice.

Plan de ce mémoire

Après une revue de la littérature existante, nous présentons les métriques et les technologies que nous proposons pour utiliser les applications existantes sur une table interactive. Nous nous attachons particulièrement à analyser les possibilités que chaque technologie offre pour modifier l'interface utilisateur des applications. Dans un deuxième temps, nous présentons l'environnement système des tables interactives et la façon dont les applications interagissent avec lui. Les informations et les événements qu'il fournit aux applications nous permettent d'identifier des interactions qui ne sont pas directement transposable du monde des ordinateurs de bureau à celui des tables interactives. Nous analysons ensuite la façon dont les tables interactives interagissent avec leurs utilisateurs. De la même manière, cela nous permet d'informer la conception des interactions que nous proposons par la suite en identifiant les interactions qui sont impossibles ou peu pratiques sur une table interactive.

Nous présentons deux scénarios d'utilisation d'applications existantes pour ordinateurs de bureau par un groupe d'utilisateur autour d'une table interactive. Ces scénarios font ressortir le besoin de techniques d'interactions spécifiques aux tables interactives pour remplacer certaines interactions existantes sur les ordinateurs de bureau. En particulier, la gestion de l'espace disponible sur la surface de la table interactive est problématique. Nous introduisons la métaphore du papier, s'inspirant de la façon dont les utilisateurs manipulent des feuilles de papier physiques. Le papier est un support d'information extrêmement répandu. En permettant aux utilisateurs d'appliquer leurs connaissances du monde physique aux interactions avec les artefacts numériques, nous pensons pouvoir augmenter l'utilisabilité des tables interactives. Nous détaillons trois techniques que nous avons plus spécifiquement étudiées : l'utilisation de fentes dans la surface de la table pour effectuer diverses action sur les documents ; le pliage non temporaire des fenêtres, qui permet de plier de façon durable une fenêtre et d'interagir avec « l'autre côté » de la fenêtre ; et le plier pour empiler, qui supporte la création impromptue de piles de documents en recyclant un document inutilisé en couverture.

Nous introduisons notre matrice des interactions pour la gestion de documents d'applications existantes. Nous nous basons sur les interactions déjà existantes dans la gestion des fenêtres sur les ordinateurs de bureau : le déplacement et le redimensionnement, auxquelles nous ajoutons trois interactions : la rotation, le zoom et le pliage. Ces interactions s'approchent de devenir des standards en matière d'interface pour table interactive dans la communauté des *tabletops* [104, 93, 10, 107]. L'orientation a été identifiée comme un point sensible de l'interaction avec les tables interactives dès 1988 [61]. Toutefois, ces interactions ne sont pas spécifiques aux tables interactives. Elles ont également été proposées dans les interfaces utilisateurs d'ordinateurs de bureau [6, 98] pour permettre d'améliorer les interactions avec des fenêtres superposées.

Les tables interactives sont des ordinateurs intrinsèquement multi-utilisateur. La disposition spatiale des utilisateurs autour d'une même table apporte naturellement des mécanismes de répartition de l'espace [105] et de synchronisme [87]. Les protocoles sociaux qui se mettent en place entre les utilisateurs déchargent le système informatique de ces tâches qui sont le focus de beaucoup de recherche dans la communauté des collecticiels sur ordinateur de bureau.

Nous concluons que tout ceci permet d'envisager l'utilisation d'applications existantes, mono-utilisateur, pour ordinateurs de bureau par plusieurs utilisateurs autour d'une table interactive.

État de l'art_____

Plusieurs travaux proposent d'ajouter de nouvelles fonctionnalités à des applications existantes. Selon l'architecture de l'application, Kacmar [57] propose plusieurs façons d'ajouter un service d'hypermédia à une application existante. Les applications basées sur un modèle d'interface séparée, ou sur un modèle d'interface fonctionnelle peuvent être augmentées avec les services d'hypermédia simplement en modifiant le modèle d'interface, en interne ou en externe. En revanche, les applications qui sont plus interconnectées à leurs modèles d'interface doivent soit être modifiées manuellement, ce qui est long et coûteux, soit être modifiées par une couche externe commune lors de leur exécution, ce que Kacmar appelle *retrofitting*. La couche externe s'interpose entre l'application, ses données et son interface pour ajouter les fonctionnalités supplémentaires.

Ces techniques permettent d'utiliser des applications existantes avec divers environnements multi-utilisateur différents, dont les tables interactives. Tse *et al.* [118] catégorisent le niveau de fonctionnalités multi-utilisateur qu'offrent ces utilisations en trois catégories : *independent applications*, *shared screen* et *true groupware*, par analogie avec la situation traditionnelle offerte par des ordinateurs de bureau individuels. Ainsi *independent applications* décrit la situation usuelle où plusieurs utilisateurs travaillent ensemble, mais chacun sur sa propre application indépendante de celle des autres. Ils utilisent des moyens de communications externes pour synchroniser leurs travaux, par exemple la voix s'ils sont dans la même pièce. En particulier, ils décrivent explicitement aux autres participants ce qu'ils voient sur leur application. Quand plusieurs utilisateurs partagent la même surface de travail, ils sont dans une situation de *shared screen*. Il peut s'agir de plusieurs personnes autour du même ordinateur, ou d'une duplication de l'affichage de la surface de travail unique sur plusieurs écrans synchronisés. Enfin, l'application utilisée peut « comprendre » que plusieurs personnes interagissent avec celle-ci, une situation que Tse *et al.* appellent *true groupware*. Elle offre plus de fonctionnalités, en particulier quand les utilisateurs travaillent en parallèle, en autorisant l'utilisation simultanée de plusieurs dispositifs d'entrée, et une navigation dans l'application et des vues indépendantes. Ils étendent ensuite leurs catégories au cas d'utilisateurs colocalisés autour d'une table interactive, en particulier en se fondant sur l'attention périphérique que les utilisateurs acquièrent dans cette situation.

Huang *et al.* [48] offrent une typologie des systèmes interactifs en fonction du nombre d'utilisateurs et du type d'espace utilisé, pour déterminer le niveau d'attention exploitable dans chaque configuration. Leur échelle du nombre d'utilisateurs va de deux utilisateurs à plus de 100, en passant par les petits groupes d'environ 10 utilisateurs. L'échelle du type d'espace contient les valeurs « ordinateur de bureau/espace personnel », « espace privé partagé » et « espace public ». Izadi *et al.* [56] ont étudié plus en détail la configuration qui nous intéresse dans le présent mémoire : une surface partagée par un petit groupe de personnes. Leurs conclusions montrent que la communication verbale entre les participants tient une place importante, en particulier pour définir les tâches de chacun. Ils donnent quelques exemples :

As the group members began to create the poster they often helped each other out by speaking aloud. For example:

L: Can anybody move this one? (has trouble grabbing title bar) I'm trying to get the old lady to the front.

(takes hand off mouse momentarily)

P grabs an occluding window and moves it out of the way

R then grabs another occluding window and moves it out of the way leaving L with space on the surface.

Other times they would ask each other to carve spaces to put media in:

R: Can someone else carve me a space and I'll put something in it

M very quickly carves a space: there's a space

C: I'll carve you the space and you can come and drop it in.

Cette communication informelle peut prendre la place de systèmes plus formels utilisés dans les applications collaboratives plus traditionnelles sur les ordinateurs de bureau. Ils relèvent également des problèmes d'interface, comme cet exemple :

However, there were also times when users in the same carved region unwittingly got in each other's way:

J makes some space for his new window by closing one in its way.

R: Oh, you closed my Events!

J: Oh Sorry, Did I?

R: That was mine that was!

J: Oops! I do beg your pardon!

Morris *et al.* [78] relèvent les mêmes problèmes de coordination entre les utilisateurs, et notent des conflits entre les interactions de chacun. Ils proposent d'introduire des mécanismes logiciels pour les résoudre, estimant que ce qu'ils nomment *social protocols* ne suffit pas pour coordonner les actions des utilisateurs. Ils définissent trois types de politiques de coordination : *proactive*, *mixed-initiative* et *reactive*, chacune pouvant s'appliquer soit à un niveau global sur l'ensemble de la table interactive, soit à un niveau local sur un élément d'interaction précis. Plusieurs exemples de tels conflits sont donnés dans [78], en relation avec le prototype développé par Morris *et al.* : le changement de la « vue » globale affichée sur la table interactive, l'utilisation d'une fonction « *magnet* » pour orienter tous les documents vers un utilisateur donné, et d'autres fonctionnalités similaires perturbent le travail des autres utilisateurs.

Les conflits créés par ces fonctionnalités pourraient aussi venir des techniques d'interaction elles-mêmes. Scott *et al.* [104] définissent huit lignes directrices pour la création de techniques d'interaction pour la collaboration colocalisée sur les tables interactives. L'une des plus importantes est de permettre aux utilisateurs de travailler en parallèle sans se gêner les uns les autres.

Dans le domaine similaire des applications collaboratives, Edwards [30] propose une autre méthode pour gérer les conflits. Les méthodes traditionnelles pour gérer les conflits sont la consistance stricte, qui empêche tout conflit via des verrous et de la prise de tour, et la résolution de conflits, qui ne permet qu'une divergence très

temporaire entre les différentes vues de l'état des données. Edwards propose de traiter les conflits comme un phénomène naturel de processus collaboratif, et de les résoudre sémantiquement, voir des les autoriser à subsister. Il utilise l'historique complet de l'état d'un objet numérique, via un système qu'il nomme *timewarp* [32], pour construire la vue commune, partagée de cet objet. Cette approche n'est pas dissimilaire de l'approche maintenant utilisée par de grands projets collaboratifs sur Internet, tel que Wikipédia, pour gérer les versions et les conflits, parfois intentionnels voire malveillants, entre les modifications simultanées de millions d'utilisateurs.

Lors de la conception d'applications pour une table interactive, la taille du groupe d'utilisateurs et de la table sont également à prendre en compte [100]. Elles changent les stratégies que développent les utilisateurs face à la même tâche collaborative. Ainsi, lors de l'étude du couplage collaboratif entre deux participants, Tang *et al.* [116] montrent que des vues indépendantes favorisent un couplage faible tandis qu'une vue globale favorise un couplage fort des utilisateurs. Le couplage est également influencé par les préférences des utilisateurs, la répartition physique des utilisateurs autour de la table, et la survenance et les mécanismes de gestion des interférences. Tang *et al.* proposent un système flexible qui s'adapte aux différentes façons de travailler des utilisateurs, et notent l'usage de protocoles sociaux pour créer des types de couplages intermédiaires entre le couplage fort et le couplage faible.

Dans ce domaine, le projet CoWord [130] utilise une approche similaire à celle que nous présenterons dans la suite de ce mémoire. CoWord permet à plusieurs utilisateurs d'utiliser le logiciel grand public Microsoft Word en même temps. Dans le protocole mis en place par Xia *et al.*, les utilisateurs sont dispersés géographiquement et utilisent chacun un ordinateur de bureau. Xia *et al.* notent qu'il leur manque des mécanismes pour supporter la conscience de l'espace collectif (*workspace awareness*) et propose de compléter le logiciel existant avec des techniques d'interaction telles que des vues radar, des télépointeurs ou des barres de défilement multi-utilisateur. Dans notre domaine des tables interactives, nous disposons de plusieurs boîtes à outils d'interaction qui implémentent des techniques adaptées au format et à la disposition des tables.

Les nouvelles difficultés d'interaction introduites par la forme physique des tables interactives peuvent aussi être résolues avec des solutions matérielles. Pour résoudre le problème de l'orientation des documents, Kakehi *et al.* [58] ont conçu une surface qui peut afficher jusqu'à quatre images différentes, selon l'angle de vue de l'utilisateur (voir figure 1). En utilisant ces quatre images pour afficher quatre vues identiques du contenu de la table à la différence près de l'orientation des textes, qui sont tournés de 90° entre chaque image, chaque utilisateur situé autour de la table peut lire le contenu de l'ensemble des documents. Pour les entrées, de manière similaire Dietz et Leigh [24] ont conçu la surface multi-tactile DiamondTouch qui permet au logiciel de savoir quel utilisateur touche la table à un endroit donné. Cette information supplémentaire sur l'identité de l'utilisateur interagissant avec un élément numérique permet de mettre en œuvre diverses interactions selon que l'utilisateur est

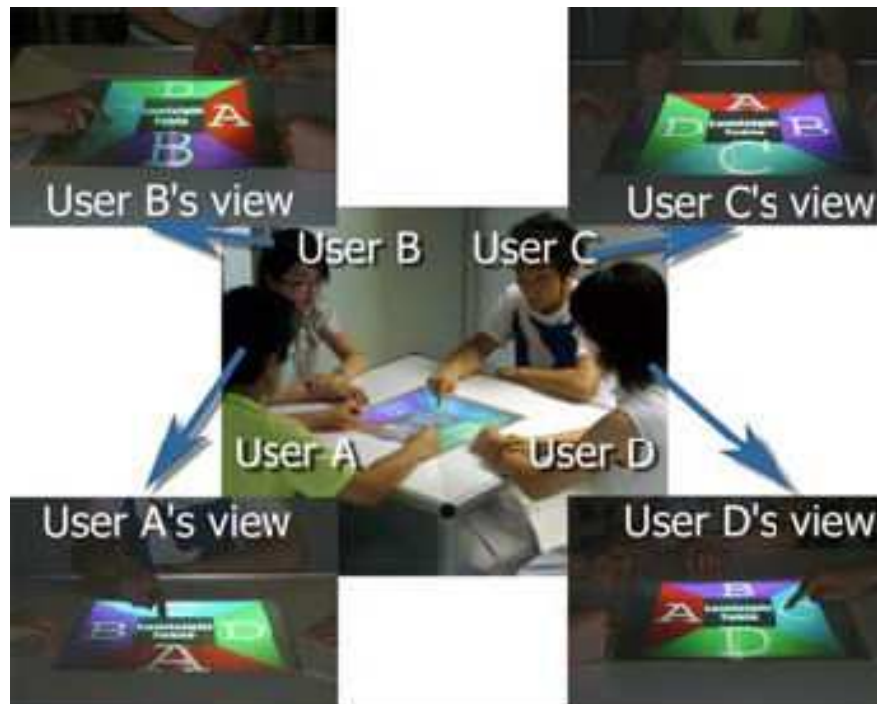


Figure 1 : La conception de cette table interactive lui permet d'afficher quatre images différentes, chacune visible uniquement depuis un angle de vue précis. Chaque utilisateur peut donc voir l'ensemble des textes correctement orientés [73].

« propriétaire » de l'objet numérique, ou selon sa relation hiérarchique avec les autres utilisateurs, ainsi que d'informer le processus de résolution de conflits.

Forlines *et al.* [37] utilisent une information supplémentaire de pression pour créer une interaction de touche à trois états : une touche avec une pression légère affiche un aperçu de l'action qui est envisagée. Celle-ci est exécutée par une touche avec une pression plus forte. Cette technique d'interaction est particulièrement adaptée à un environnement partagé par plusieurs utilisateurs, en augmentant à la fois la découvrabilité des conséquences de ses propres actions et la conscience de ce que prévoient de faire les autres utilisateurs, elle permet de résoudre les conflits d'interaction avant qu'ils ne se produisent.

Les tables interactives apportent par ailleurs de sérieux avantages dans le domaine de la collaboration. L'utilisation d'ordinateurs individuels dans une situation de collaboration est apparue peu pratique depuis longtemps. Dès 1985, Stefik *et al.* [111] proposaient d'équiper des salles de réunion de grands écrans tactiles partagés et de logiciels spécifiquement conçus pour supporter la collaboration. [111] ne propose pas de résultats quantifiables de l'impact de leur technologie sur l'activité des utilisateurs.

En partenariat avec l'ENSAM, nous avons mené une étude [12, 13, 83] sur l'impact des tables interactives sur une tâche de collaboration bien définie : la créativité. « *Creativity is the ability to produce work that is : novel (i.e. original, unexpected), high in quality, and appropriate (i.e. useful, meets task constraints)* » a écrit le psychométricien Robert Sternberg [112]. Les tâches de créativité font l'objet de recherche dans le domaine de la psychologie et y sont définies de manière très

précise. Nous avons mené l'étude sur deux de ces tâches collaboratives, la carte mentale et la tâche de purge. Pour chaque tâche, nous avons comparé dans des expériences contrôlées la situation traditionnelle où les utilisateurs se servent d'un tableau blanc et d'accessoires de papeterie, avec la situation de table interactive, où la table propose une zone de travail partagée et des artéfacts numériques correspondant aux accessoires de papeterie (encre virtuelle, post-it virtuels, etc.). Les résultats montrent que la collaboration autour d'une table interactive a été jugée plus agréable par les utilisateurs, et rend la communication entre utilisateurs plus plaisante, toutefois sans augmenter la productivité globale du groupe. En revanche, l'utilisation de la table interactive a permis de réduire significativement le phénomène négatif de paresse sociale (*social loafing*) qui se produit naturellement au sein d'un groupe de personnes dans ces situations de travail collaboratif. La paresse sociale se traduit par un déséquilibre dans la quantité de contribution des membres du groupe, et l'apparition de « leaders » et de « suiveurs ». Nous avons observé que l'utilisation d'une table interactive rééquilibre les contributions entre les membres du groupe. Cette observation pourrait être une conséquence de l'agréabilité de la communication entre utilisateurs, qui stimulerait la créativité en favorisant l'échange d'idées.

Rogers *et al.* [96] ont mis en évidence un effet similaire des tables interactives sur le processus de prise de décision collaborative. La prise de décision lors de réunions en face-à-face (i.e. colocalisées), Rogers *et al.* notent [97], se fait sur la base de données externes ce qui introduit une asymétrie entre les participants. S'il est possible, avec les technologies actuellement utilisées dans ces situations, telles que les tableaux blancs ou les vidéo projecteurs, d'obtenir un accès égal à l'information affichée, il est nettement plus difficile d'obtenir un accès égal à la manipulation et à la création de ces données. En particulier, une personne peut monopoliser les moyens d'interactions (e.g. marqueur Velléda, clavier et souris, etc.) avec les données affichées sur les supports partagés. Des précédents travaux montrent que lorsqu'une personne obtient le rôle de contrôleur de l'interaction, elle a tendance à le conserver. Les autres participants peuvent alors trouver plus difficile d'influer sur les données, et le processus de décision, si leurs suggestions ou leurs idées sont ignorées par cette personne.

L'étude de Rogers *et al.* porte sur la construction collaborative d'un calendrier. Trois utilisateurs doivent se mettre d'accord sur le choix de douze photographies pour représenter les douze mois de l'année. Les utilisateurs utilisent une table multi-touche, ce qui leur permet d'interagir simultanément en parallèle. L'analyse des comportements de décision (questionnement, instruction, suggestion, offre et confirmation) met en évidence une plus grande participation de l'ensemble des membres du groupe. De manière intéressante, les participants ont peu interagi en parallèle, bien qu'ils le pouvaient, optant à la place pour un système de tours informels. Il y a donc un « contrôleur de la table », mais celui-ci ne conserve pas son rôle plus que le temps nécessaire pour appliquer son idée ou sa suggestion.

En dépit de ces avantages, les applications utilisables sur des tables interactives sont en nombre limité. La plupart sont des prototypes de recherche, telles que les deux applications que nous avons développées pour l'expérimentation en partenariat avec

l'ENSAM précitée, et ne disposent pas de toutes les fonctionnalités que les utilisateurs attendent. Nous nous trouvons dans une situation similaire à celle dans laquelle la communauté du travail coopératif assisté par ordinateur (CSCW, *computer-supported cooperative work*) se trouvait il y a plusieurs années. Les travaux dans ce domaine se sont orientés rapidement vers l'utilisation d'applications existantes. Dès 1975, Engelbart [34] explorait l'utilisation de l'application existante NLS Journal sur un écran partagé par plusieurs utilisateurs via Arpanet. Plusieurs autres travaux similaires ont suivi [38, 62, 22, 1, 40]. L'utilisation d'applications existantes a permis à la fois d'augmenter le nombre d'applications compatibles avec les systèmes de travail coopératif développés par la communauté CSCW, mais aussi à cette communauté de se concentrer sur les problèmes spécifiques au travail coopératif assisté par ordinateur [63, 64], plutôt que sur le développement d'applications.

Nous proposons d'utiliser une démarche dans le même esprit pour augmenter le nombre d'applications utilisables sur les tables interactives. Nous explorons ensuite quelques problèmes d'interaction homme-machine qui sont posées par cette utilisation.

En effet, l'ordinateur de bureau a toujours été conçu avec une unique souris et un unique clavier. Dans les années 70, les chercheurs du Xerox PARC travaillaient et vivaient dans l'environnement futuriste qu'ils avaient créé, tel que Colab [111], une salle de réunion informatisée expérimentale équipée d'ordinateurs et de tableaux tactiles pour utiliser coopérativement des bases de données distribuées. Pourtant, quand ils ont conçu le premier ordinateur personnel, le Alto (voir figure 2), ils n'ont inclus qu'un seul clavier et une seule souris. Si certaines caractéristiques du Xerox Alto, comme son écran au format portrait, ont depuis longtemps été reléguées au rang des curiosités historiques, cette unique souris et cet unique clavier sont depuis devenus la norme en matière d'ordinateur de bureau [113].



Figure 2 : Le Xerox Alto, le premier ordinateur personnel à utiliser une interface graphique, ne peut utiliser qu'un unique clavier et une unique souris. Cette caractéristique est depuis la norme pour les ordinateurs de bureau.

Nous observons, ces derniers temps, une multiplication de plusieurs des périphériques de cet ordinateur du bureau : les systèmes multi-écran sont maintenant courant, plusieurs cartes réseaux sont pratiquement la norme, et plus personne ne penserait à avoir de l'audio sans au minimum deux canaux en stéréo. Mais les périphériques d'entrée restent limités à un clavier et à une souris, ou plus exactement à un pointeur de souris [65]. Les styles d'interactions homme machine disponibles dans les applications pour ordinateurs de bureau sont construits sur ce principe fondamental de l'unicité du clavier et de la souris. Pourtant, l'utilisation de plusieurs souris permet d'augmenter les performances lors d'activités de groupe [55].

Le format même des tables interactives, horizontal, de grande taille, pose des problèmes au créateur d'interfaces utilisateur. Par exemple, les éléments de l'interface peuvent être hors de portée de l'utilisateur parce qu'ils sont situés de l'autre côté de la table par rapport à lui. Les utilisateurs, placés tout autour de la table, peuvent voir l'interface et son texte à l'envers ou selon un angle où elle est difficilement lisible [123]. Les composants standards des boîtes à outils d'interface utilisateur peuvent devenir plus difficiles à utiliser sur une table. Scott *et al.* [102] donnent l'exemple du composant probablement le plus utilisé dans les interfaces utilisateur des applications pour ordinateur du bureau : le bouton. Le bouton standard, dans la plupart des boîtes à outils, utilise un effet graphique en fausse 3D, introduit en 1989, pour indiquer son état (désactivé, prêt, pressé, etc.). Cet effet devient ambigu quand il est vu à l'envers (figure 3).

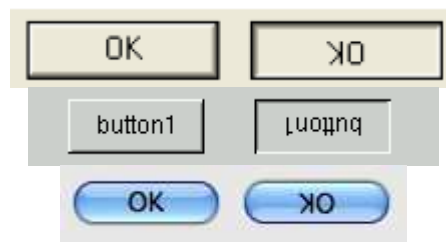


Figure 3 : Le composant standard « bouton » devient ambigu quand il est observé à l'envers (haut : boîte à outils de Windows XP, milieu : boîte à outils GTK+). Le bouton de la boîte à outils Aqua de MacOS X (bas) résiste mieux à ce changement d'orientation.

Dans le premier chapitre de cette thèse, nous nous basons sur les travaux qui proposent d'ajouter de nouvelles fonctionnalités à des applications existantes pour proposer des solutions pour utiliser ces applications sur des tables interactives. Dans les chapitres suivants, nous explorons les problèmes d'utilisabilité que cela pose, et proposons des techniques d'interactions pour les résoudre.

CHAPITRE 1

Amener les applications existantes sur les tables interactives

1.1. Introduction

L'utilisation d'applications existantes sur les tables interactives commence avec le problème de pouvoir faire utiliser l'environnement système de la table à l'application. En effet, les tables interactives ne sont pas des ordinateurs de bureau, elles ont des fonctionnalités différentes, et donc les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau. Dans ce premier chapitre, nous nous proposons d'examiner différentes solutions techniques permettant d'établir un lien entre une application existante et l'environnement d'une table interactive.

Ce lien est essentiellement caractérisé par sa position dans la chaîne de traitement entre l'application et le trio de dispositifs écran-clavier-souris pour lesquels elle a été conçue. La figure 4 présente le schéma simplifié de l'architecture type d'un système exploitation pour ordinateur de bureau. Différentes possibilités de liens entre l'application et le système de table interactive sont représentées de différentes couleurs. Les détails sur le fonctionnement de ces divers liens sont donnés dans leurs descriptions respectives après la présentation des axes de comparaison.

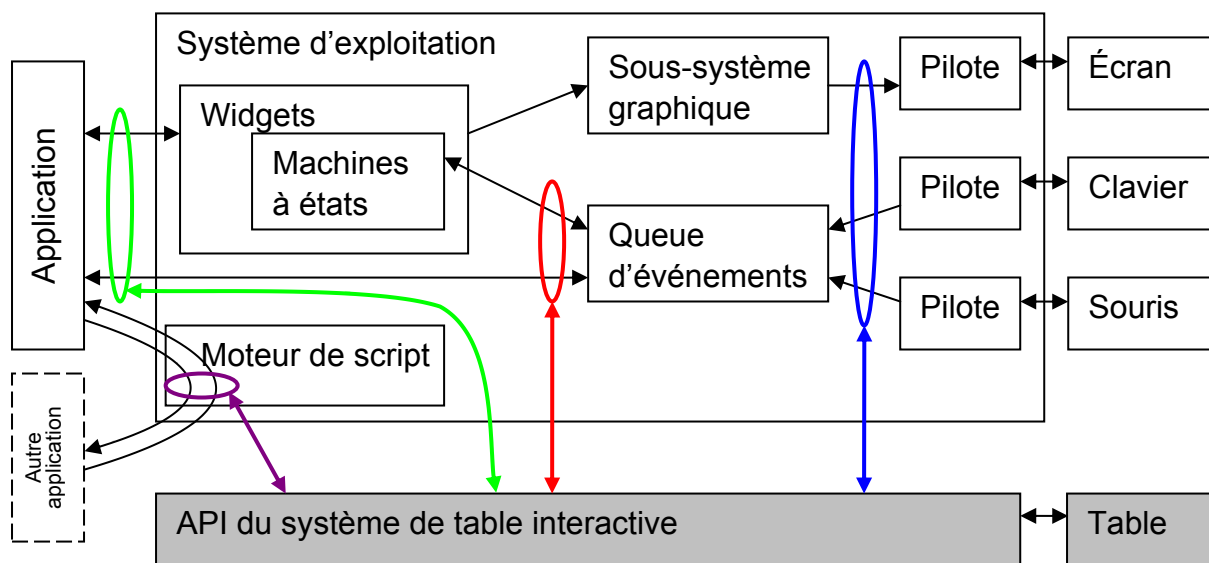


Figure 4 : Un système d'exploitation de bureau permet à une application (à gauche) d'interagir avec un utilisateur via un clavier, un écran et une souris (à droite). Le système de table interactive (en bas) vient se greffer (flèches en couleur) sur les liens de communication (flèches noires) entre les diverses parties du système d'exploitation et/ou les applications.

Le bon positionnement de ce lien est un défi complexe : il faut arriver à équilibrer les fonctionnalités qui pourront être obtenues, les performances de l'ensemble application-lien-table, et la quantité de « glue » nécessaire pour faire « coller » le lien.

Nous avons identifié un certain nombre d'approches pour créer un lien entre une application existante non modifiable et un système de table interactive. Dans toutes

ces approches, le système interagit avec les applications d'une manière originale, non prévue par les développeurs originaux de l'application. La principale différence entre ces technologies est la position et la nature du lien ainsi créé, qui influent sur les performances et les fonctionnalités qui pourront être obtenues. Pour pouvoir les comparer, nous avons identifié plusieurs axes pour caractériser ces liens.

Nos cinq axes permettent chacun d'évaluer un aspect des conséquences d'une technologie de lien sur le phénomène d'interaction entre les utilisateurs d'une table interactive et une application pour ordinateur de bureau. Des performances perçues du système au support d'interactions multi-point, les différentes technologies disponibles ont toutes des avantages et des inconvénients. Un tableau résume l'ensemble des axes et des technologies à la fin de ce chapitre.

1.2.1. Structuration des données

Le premier axe est la structuration des données. Il s'agit du niveau de structuration des données échangées entre l'application et le système de table interactive. Les valeurs de cet axe vont de « non structuré » à « très structuré ». Des données structurées peuvent se présenter par exemple sous la forme d'une description de chaque composant de l'interface (fenêtres, boutons, zones de texte, etc.) et de leurs relations entre eux. Des données très peu structurées, au contraire, pourraient être une ou plusieurs images de type bitmap des fenêtres de l'application, telles qu'elles seraient affichées sur un écran d'ordinateur de bureau. Des données structurées sont importantes parce qu'elles permettent plus facilement les manipulations par un programme informatique. Elles permettent un contrôle plus fin de l'interaction et des modifications plus radicales de l'interface utilisateur de l'application. Par exemple, une description structurée d'une interface permettrait au lien logiciel entre l'application existante et le système de table interactive de transformer automatiquement toutes les boîtes de propriétés à onglets de l'application par une métaphore de livre dont les pages se tournent d'un geste de la main.

La structuration des données concerne aussi les dispositifs en entrée. Des données en entrée structurées utilisent des commandes de haut niveau, soit orientées interaction, soit orientées action. Dans le cas d'une commande de haut niveau orientée interaction, le lien envoie, par exemple, à l'application existante un événement « ce menu a été sélectionné » sans justifier l'origine de cette sélection dans un clic de souris, un raccourci clavier, une commande vocale, ou notre dispositif de table interactive. Dans le cas d'une commande de haut niveau orientée action, le lien invoque, par exemple, la commande « charger un fichier » de l'application. Les deux cas nécessitent la présence d'informations suffisamment détaillées sur l'application, son interface et/ou les actions qu'elle supporte, et le format de données qu'elle utilise. Des données en sortie suffisamment structurées apportent ces informations. Les données en entrée non structurées, au contraire, sont de bas niveau d'abstraction, comme par exemple simuler le périphérique d'entrée pour lequel l'application a été développée (le plus souvent un clavier et une unique souris).

1.2.2. Flexibilité des modifications possibles

Le second axe est la flexibilité des modifications possibles offertes par le lien. Comme nous l'avons déjà dit, les systèmes de tables interactives ont des caractéristiques très différentes des ordinateurs de bureau pour lesquels les applications ont été conçues. Les systèmes de sortie graphique et d'entrée de l'application doivent être modifiés ou remplacés pour correspondre à celui des tables interactives. Les différentes façons de lier les applications à une table interactive offrent plus ou moins de flexibilité pour faire ces modifications. Ainsi, par exemple, pour permettre de réorganiser librement les fenêtres sur la surface de la table, il faut que le lien force l'application à dessiner la totalité du contenu de ses fenêtres, y compris les zones masquées par d'autres fenêtres dans une configuration d'ordinateur de bureau. S'il ne le fait pas, le développeur est limité dans la façon dont il peut réorganiser les fenêtres.

De façon similaire, les tables interactives permettent à plusieurs personnes d'interagir en même temps, ou à un même utilisateur d'utiliser ses deux mains pour interagir avec deux éléments de l'application en parallèle. Les applications pour ordinateurs de bureau sont conçues avec des boîtes à outils qui n'utilisent qu'un seul curseur de souris, qui ne définit qu'un seul focus sur un seul composant à la fois. Ce composant est celui qui reçoit les événements du clavier (par exemple avec deux champs de texte, la souris permet de donner le focus à un seul d'entre eux, qui sera celui dans lequel le texte saisi s'ajoutera). Pour tirer le meilleur parti des dispositifs d'entrée multi-utilisateur et multi-point que sont les tables interactives, il est nécessaire dans de nombreux cas que le lien puisse changer ce comportement, et les états internes des composants de l'interface utilisateur.

Les applications utilisées elles-mêmes peuvent avoir un impact sur le degré de flexibilité atteignable. Par exemple, peut-être une application donnée supporte-t-elle de recevoir deux événements « le bouton de la souris a été pressé » à deux endroits différents, même si elle n'a pas été conçue pour gérer cette situation qu'il est impossible de produire avec une souris d'ordinateur de bureau. Pour avoir le maximum de flexibilité avec le maximum d'application, un haut niveau de structure est néanmoins nécessaire pour pouvoir agir plus finement sur une partie précise des systèmes de sortie graphique et d'entrée.

1.2.3. Performance

Le troisième axe est la performance de l'ensemble du système. Les systèmes interactifs doivent être rapides à réagir à un événement utilisateur et à mettre à jour leur affichage en conséquence [70]. Le principal paramètre est la quantité de traitements que le lien et le système de table interactive doivent effectuer en plus de ceux effectués par l'application, et la quantité de traitements effectués par l'application, mais inutiles car inadaptés aux tables interactives. Dans le pire des cas, le lien et le système de table interactive doivent refaire tous les traitements normalement faits par l'application, mais l'application effectue quand même ces traitements elle-même, et leurs résultats sont simplement ignorés. Dans le meilleur des cas, le lien est positionné dans la chaîne de traitement (voir figure 4) de telle manière qu'aucun traitement ne soit fait en double. Le temps de communication

entre l'application et le système de table interactive doit également être optimisé. Il peut devenir important si une grande quantité de données doit être convertie entre plusieurs formats de données incompatibles, ou doit être transmise par un réseau informatique.

1.2.4. Compatibilité et réutilisabilité

Le quatrième axe est la compatibilité et la réutilisabilité. La compatibilité avec les applications détermine l'intérêt global d'une technologie de lien en particulier. En effet, nous voulons pouvoir utiliser le plus d'applications existantes possible sur une table interactive. Un lien qui ne serait compatible qu'avec certaines applications, par exemple dépendant des outils qui ont été utilisés pour créer l'application, présenterait un intérêt inférieur à un système fonctionnant universellement avec toutes les applications. La réutilisabilité détermine l'utilité d'un prototype. Certaines technologies de liens peuvent avoir besoin que chaque application à prendre en charge soit spécifiquement prévue (par exemple via des plug-ins programmés pour chaque application), tandis que d'autres technologies de liens fonctionnent directement avec toutes les applications. Ces dernières sont plus utiles dans l'immédiat, car elles permettent d'explorer plus de scénarios d'utilisation des applications sur une table interactive, sans devoir redévelopper un morceau de logiciel spécifique pour chacune.

1.2.5. Difficulté d'implémentation

Enfin, le cinquième axe est la difficulté d'implémentation du lien logiciel. Certaines techniques requièrent peu de travail d'implémentation pour obtenir un prototype fonctionnel, puis proportionnellement plus de travail pour ajouter plus de fonctionnalités. D'autres en revanche requièrent une grande quantité de travail d'implémentation avant d'avoir un premier prototype, mais permettent d'obtenir directement un produit quasiment complet. Bien sûr, certaines technologies pourraient aussi demander peu de travail pour obtenir un produit fini.

1.3. Technologies

Nous avons présenté cinq axes pour comparer les technologies qui permettent d'utiliser une application existante avec un environnement système différent : celui d'une table interactive. Nous allons maintenant évaluer différentes technologies selon ces critères de structuration des données, flexibilité, performance, compatibilité et réutilisabilité, et difficulté d'implémentation.

Nous avons étudié en détails six technologies, et implémenter des prototypes pour trois d'entre elles. Les descriptions des implémentations et leurs résultats sont détaillées dans les sections correspondantes. Un tableau résume l'ensemble des axes et des technologies à la fin de ce chapitre.

1.3.1. Capture d'écran

La méthode la plus simple pour obtenir la sortie graphique d'une application est sans aucun doute de prendre une capture d'écran à intervalle régulier. La capture d'écran enregistre une copie exacte de l'image affichée sur l'écran de l'ordinateur dans une zone mémoire où elle peut être modifiée pour être adaptée à l'environnement de la table interactive. L'image de l'écran ne comporte aucune métadonnées supplémentaires, mais tous les systèmes d'exploitation pour ordinateurs de bureau que nous connaissons offrent la possibilité de connaître la taille et la position des fenêtres affichées à l'écran. Ces informations peuvent être utilisées pour segmenter la capture d'écran en images de chaque fenêtre afin de les gérer séparément. Toutefois, ce sont les seules informations que l'on peut obtenir sur l'application utilisée. Cette technologie a donc un très faible niveau de structuration des données. Elle manque aussi de flexibilité. Ainsi, si deux fenêtres sont superposées dans la configuration pour l'écran d'ordinateur de bureau, les parties masquées ne pourront pas être récupérées par la capture d'écran. Nous serons donc limités dans la façon dont on peut réorganiser les fenêtres.

Une approche similaire a été utilisée pour interagir avec des applications d'ordinateur de bureau depuis un périphérique portable [80]. L'image de l'écran de l'ordinateur de bureau est affichée en entier sur le périphérique portable, avec un niveau de zoom variable. Des informations supplémentaires sont utilisées pour adapter les menus des applications et les zones de texte au format du périphérique. Ainsi, les menus d'origine sont supprimés et remplacés par des menus multi colonne. Les textes des zones de texte sont quand à eux reformatés pour avoir des lignes plus courtes qui tiennent dans la largeur de l'écran du périphérique portable [79].

1.3.2. Carte graphique ou écran virtuels

Une amélioration de cette première technique consiste à utiliser une carte graphique virtuelle ou un écran virtuel inséré dans l'environnement de l'ordinateur de bureau. C'est l'approche utilisée par la plupart des logiciels de contrôle à distance d'ordinateur (par exemple VNC [92] installe un pilote de carte graphique virtuelle « VNC hook » sous Windows, et crée des écrans X11 virtuels sous Unix).

Nous avons mis en œuvre cette technique pour utiliser des applications X Window existantes avec l'environnement de développement pour table interactive DiamondSpin [108]. Le X Window System [101] est le logiciel couramment utilisé pour gérer l'interaction homme-machine par l'écran, la souris et le clavier sous Linux et les diverses variantes d'UNIX. Sur un ordinateur de bureau, le logiciel serveur X communique avec l'écran, le clavier et la souris. Il reçoit et sert des requêtes d'affichage, d'entrées de texte et de déplacement de souris, qui proviennent des logiciels client X (les applications). Elles se connectent au serveur X et lui envoient leurs requêtes d'affichage en utilisant le protocole X.

Nous utilisons le logiciel Metisse [99] pour communiquer avec les applications. Metisse est basé sur X Window, mais au lieu de faire le lien entre les applications et l'écran, le clavier, et la souris, le serveur Metisse stocke les fenêtres à afficher sous forme d'images numériques [17]. Le serveur Metisse est un serveur X modifié pour dessiner les fenêtres hors écran, rendre leurs images accessibles à un compositeur et rediriger les événements souris/clavier envoyés par ce compositeur vers les applications (figure 5). Les applications n'ont pas besoin d'être modifiées ou recompilées, et ne savent pas qu'elles ont été redirigées.

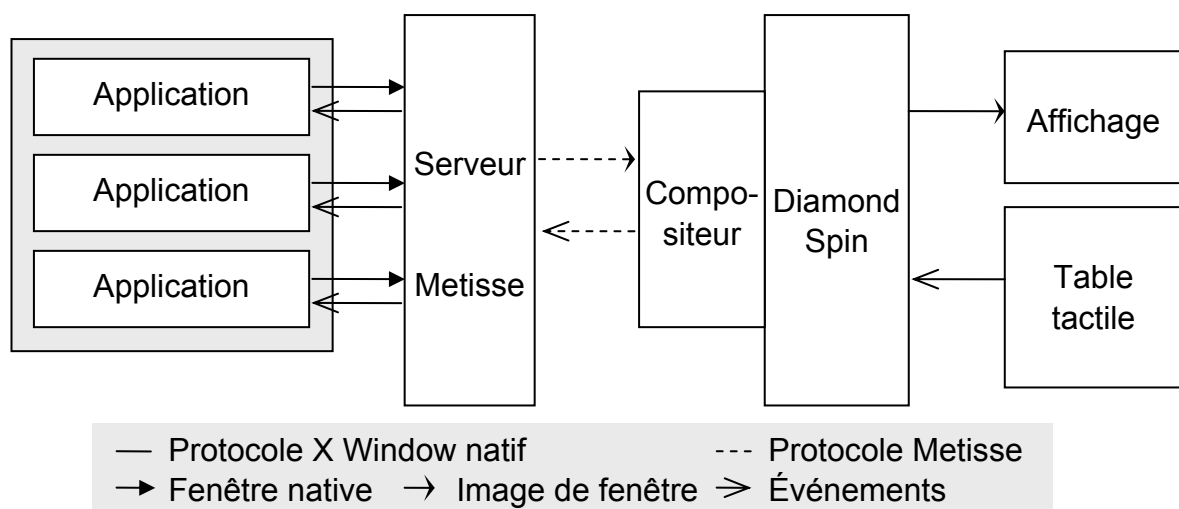


Figure 5 : Architecture du système, incluant Metisse et DiamondSpin, pour utiliser des applications X Window existantes avec une table interactive.

Le serveur X de Metisse rend accessible les images des fenêtres par une connexion réseau. Nous avons donc implémenté un module client pour DiamondSpin. Du point de vue du serveur Metisse, notre module est un compositeur qui va récupérer les images des fenêtres et générer des événements clavier et souris. Le protocole de communication entre le serveur Metisse et le compositeur est dérivé du protocole RFB : les mises à jour des fenêtres (et non pas seulement du Frame Buffer) sont envoyées sous la forme d'images rectangulaires correspondants à la plus petite zone englobant la modification. Ces images sont récupérées et stockées par DiamondSpin dans des textures OpenGL, qui les positionne ensuite de manière appropriée par

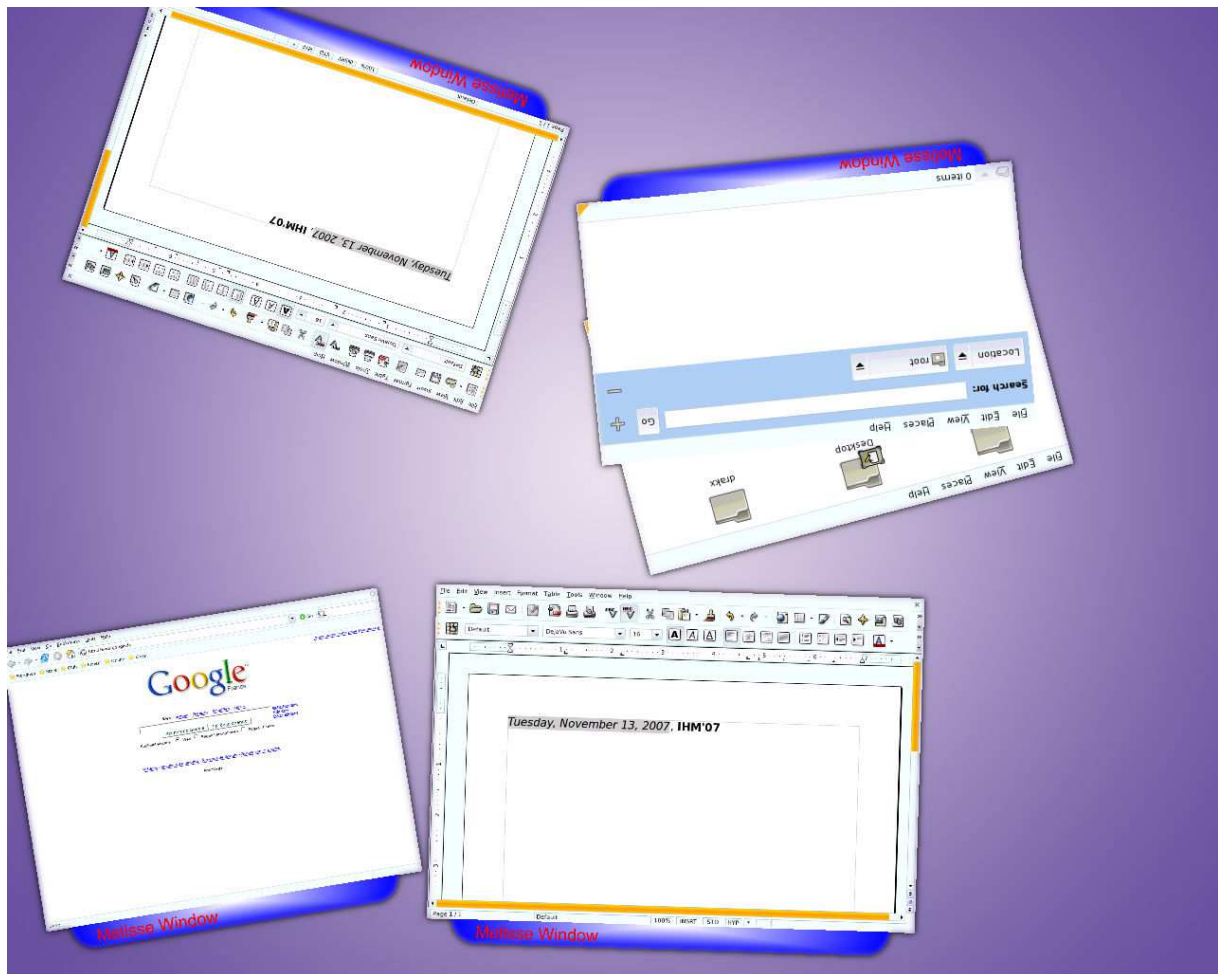


Figure 6 : Plusieurs applications Linux existantes (Firefox, OpenOffice, un explorateur de fichier) utilisées sur une table interactive. La fenêtre OpenOffice est dupliquée pour permettre à deux utilisateurs d'éditer le document.

rapport aux utilisateurs (orientées dans le sens de lecture de l'utilisateur qui manipule la fenêtre,...) et les affiche sur la table (figure 6).

Cette technologie a un niveau de structuration des données légèrement plus élevé que la précédente. Chaque fenêtre est gérée séparément, et des méta données, telles que la taille ou la position des fenêtres, sont fournies et mises à jour automatiquement via le protocole Metisse. Toutefois l'image de chaque fenêtre reste une simple image numérique bidimensionnelle. Cette technologie offre aussi plus de flexibilité, notamment pour la réorganisation des fenêtres car la situation d'occlusion décrite précédemment n'existe plus ici. Les méta informations sur chaque fenêtre nous ont aussi permis d'adapter quelques éléments d'interface. Par exemple, nous remplaçons à la volée les barres de titre des fenêtres par des barres de titre plus grosses, pour une manipulation avec le doigt plus facile, et situées en bas des fenêtres, pour éviter que la main de l'utilisateur ne cache le contenu quand il manipule la fenêtre par sa barre de titre. Plus de détails sur ce dernier point sont disponibles dans le chapitre 4.

Au niveau des performances, cette technique permet un taux de rafraîchissement de l’affichage élevé pour les applications de bureau typiques que nous avons testé.

1.3.3. Clavier virtuel et une souris virtuelle

Les captures d’écran et les cartes graphiques ou écrans virtuels sont toutes les deux des technologies pour obtenir le rendu graphique d’une application, mais elles ne permettent pas d’envoyer des événements aux applications. Cette troisième technologie est leur équivalent pour les entrées clavier et souris : simuler un clavier virtuel et une souris virtuelle. Elle est intégrée dans le système Metisse. Un ordinateur de bureau traditionnel est limité à une seule souris et un seul clavier (ou plus exactement un seul pointeur de souris et un seul focus de clavier) partagés par l’ensemble des applications.

Plusieurs projets permettent d’utiliser des applications existantes avec plusieurs pointeurs en parallèle, mais les niveaux de fonctionnalités varient. Le Transparent Input Device Layer [53] se propose d’utiliser plusieurs pointeurs avec des applications Java existantes pour annoter collaborativement les données de ces applications. En revanche, ce système ne supporte pas l’interaction avec l’application elle-même : une couche intermédiaire intercepte les événements à destination de l’application et les interprète comme des annotations. Westergaard [124] a implémenté un pilote de périphérique pour utiliser plusieurs souris physiques contrôlant plusieurs pointeurs à l’écran sous Microsoft Windows, mais les applications doivent être modifiées et utiliser une API spécifique pour accéder à plus d’un curseur. Le Multi-Pointer X Server (MPX) [54] autorise plusieurs pointeurs de souris pour les opérations de gestion de fenêtres (déplacement, redimensionnement, etc.), mais, à nouveau, seules les applications spécifiquement modifiées peuvent utiliser plus d’un pointeur à la fois pour les autres opérations (éditions de texte, sélection, dessin, etc.).

L’unicité du focus du clavier impose que, à un instant donné, un seul élément de l’interface reçoive les événements du clavier. Pour les systèmes utilisant plusieurs pointeurs de souris dans le cadre d’une interaction bi-manuelle sur un ordinateur de bureau, cette restriction n’est pas excessivement contraignante : l’unique utilisateur de l’ordinateur du bureau n’a de toute façon qu’un seul clavier. En revanche, pour les systèmes multi-utilisateur comme les tables interactives, plusieurs utilisateurs peuvent vouloir saisir du texte dans différentes applications en même temps.

D’autre part, certains éléments du système sont modaux, soit au niveau d’une application (par exemple, une boîte de dialogue modale), soit au niveau de l’ensemble du système (par exemple, un menu surgissant). Sur une table interactive, plusieurs utilisateurs interagissent en même temps avec le système et les interactions modales sont une gêne importante. Une gestion spécifique du focus est nécessaire pour rendre ceci transparent aux applications redirigées. Cette gestion peut également découpler le focus du clavier de celui de la souris.

Par exemple, Metisse permet de spécifier quelle fenêtre doit recevoir un événement souris ou clavier. Cela permet au minimum à plusieurs utilisateurs d'interagir chacun avec une application différente. En effet, du point des applications redirigées, il n'y a qu'un seul curseur : toutes les applications du même terminal pensent se partager une unique souris entre elles. Quand l'utilisateur touche brièvement la table (l'équivalent d'un clic), le curseur, du point de vue de l'application, se déplace à la vitesse maximale vers la zone touchée, et effectue un clic. Le curseur se déplace ensuite à la prochaine zone touchée, éventuellement par un autre utilisateur. Ce mécanisme limite toute opération de type glisser-déposer, puisque pendant le glisser, le bouton de la souris virtuelle doit rester pressé. Il est donc indisponible pour tout autre « clic » multitouche, ce qui effectivement verrouille la table pendant l'interaction de glisser-déposer.

Nous pouvons également créer un trio écran virtuel, souris virtuelle, clavier virtuel par application, chaque application a alors le contrôle complet de son terminal informatique virtuel. La création d'un trio de périphériques écran-clavier-souris virtuels par application permet à chaque application de disposer de son propre curseur de souris. Les utilisateurs de la table interactive peuvent ainsi faire plusieurs glisser-déposer en parallèle, mais ces glisser-déposer sont dans ce cas limités à être effectués entre deux fenêtres de la même application (i.e. du même terminal virtuel).

1.3.4. Langage de script

Une philosophie complètement différente est mise en œuvre pour cette autre solution permettant de lier une application existante avec un environnement de table interactive : l'utilisation d'un langage de script intégré à l'application ou au système d'exploitation. Certaines applications intègrent un langage de script, tel que VBA (Visual Basic for Application) ou AppleScript, dans le but de faciliter l'automatisation des tâches courantes de l'utilisateur. Ces langages peuvent se placer aux niveaux des données de l'application (un accès direct aux fonctions du modèle dans le schéma modèle-vue-contrôleur). Par exemple, un script VBA peut ouvrir un fichier du tableur Excel, itérer sur les valeurs contenues dans une colonne et modifier les formules d'autres cellules. Certaines applications autorisent des applications tierces à accéder à leur état interne via le langage de script. Par exemple, un navigateur web incluant un langage de script peut être utilisé dans le scénario suivant. Une application tierce, externe, peut appeler une fonction de script pour que le navigateur web se connecte à une page spécifique, puis récupérer des morceaux de la page web comme l'arbre DOM (*Document Object Model*) du document.

Dans le cadre de l'utilisation d'applications existantes sur une table interactive, nous pouvons utiliser ce système de langage de script pour accéder aux données d'une application existante sans qu'elles soient formatées pour l'affichage sur un ordinateur du bureau. Par contre, nous sommes alors responsables de l'implémentation de ces fonctions de formatages des données pour l'affichage et des fonctions d'interaction de l'utilisateur de la table avec les données (la vue et le contrôleur dans le modèle MVC). C'est à la fois une opportunité très intéressante puisque cela permet d'afficher et d'interagir avec les données de la façon la mieux adaptée au système de table

interactive : le multi-utilisateur et le multi-touche ne sont plus des problèmes mais deviennent des fonctionnalités intégrantes de l'application, les données peuvent être zoomées de façon appropriée en fonction de leur position sur la table, ou dupliquées pour chaque utilisateur tout en maintenant des vues indépendants pour chacun. D'un autre côté, cela veut aussi dire que chaque application devra être spécifiquement scriptée, puisque chacune a des données différentes à présenter sous un format différent. Un certain nombre de fonctionnalités de bases sont standardisées dans certains langages de script, comme charger un fichier. On peut ainsi utiliser une boîte de dialogue « sélection du fichier » programmée spécifiquement pour les tables interactives, puis appeler la fonction « charger un fichier » d'une application en lui passant directement le nom du fichier. Voire ne plus utiliser la métaphore des fichiers et des dossiers, mais utiliser un interacteur tangible pour représenter un document, et appeler la fonction « charger un fichier » avec le nom du fichier dans lequel est stocké le document. Certains systèmes d'exploitation ont une standardisation plus avancée des fonctionnalités de scripting disponibles dans leurs applications. Ainsi, de nombreuses applications pour MacOS X utilise le langage AppleScript [3] introduit par Apple en 1992. Mais la plupart des opérations plus complexes sont spécifiques à chaque application, ce qui fait que la technologie de scripting a une réutilisabilité très faible d'une application à l'autre. De plus, toutes les applications ne disposent pas d'un langage de script, la technologie de scripting a donc aussi une faible compatibilité.

En revanche, la technologie de scripting est performante car toutes les données échangées avec les applications sont sous la forme de leur représentation interne, ce qui est beaucoup plus léger que la représentation graphique des données sur un écran à travers une vue donnée. L'affichage et les interactions avec l'utilisateur sont gérés nativement par le système de table interactive, leurs performances sont donc maximales.

Nous avons implémenté un tel système pour utiliser Microsoft Excel à plusieurs utilisateurs simultanément autour d'une table interactive. Excel, comme le reste de la suite Microsoft Office, utilise le système de communication inter-processus Automation. Automation est un mécanisme basé sur COM (*Component Object Model*), qui permet à une application, appelée serveur, d'exposer des objets, composés de propriétés et de méthodes, à d'autres applications, appelées des contrôleurs. En particulier, le langage de script VBA (*Visual Basic for Applications*) est un contrôleur Automation natif, et dispose d'une syntaxe spécifique pour manipuler les objets Automation.

Excel expose sous forme d'objets Automation son modèle de données. Ainsi l'objet *Workbook* correspond à un fichier. Il contient des objets *Worksheet* correspondant à une feuille Excel, qui contiennent des objets *Range* correspondant aux cellules de la feuille. Les cellules contiennent des propriétés comme la valeur numérique de la cellule, la formule utilisée pour la calculer, la taille, la police de caractères, la couleur etc. de la cellule.

Nous avons implémenté un contrôleur Automation en C++ et Java qui utilise les objets Automation exposés par Excel pour gérer son modèle de données. Les interactions avec les utilisateurs sont gérées par la boîte à outils pour tables interactives DiamondSpin. Quand l'utilisateur modifie un élément du fichier Excel, notre contrôleur Automation répercute ce changement sur les objets Automation. Microsoft Excel se charge de mettre à jour les autres cellules en utilisant ses propres mécanismes. Les aspects multi-utilisateur et spécifiques aux tables interactive, tels que de multiples focus d'entrée clavier, l'orientation des documents, ou l'affichage de plusieurs parties du document par plusieurs utilisateurs en parallèle sont également gérés avec par DiamondSpin. Le contenu de cellules, leurs tailles, la police de caractères, et les autres données nécessaires à l'affichage sont extraites des objets Automation les représentant (figure 7) et utilisées par un programme d'affichage en OpenGL. Ce programme a été développé spécifiquement pour afficher des données provenant d'Excel. Cette approche ne s'applique qu'aux applications existantes non modifiables spécifiquement prévues par le développeur de la couche de liaison entre les applications et le système de table interactive.

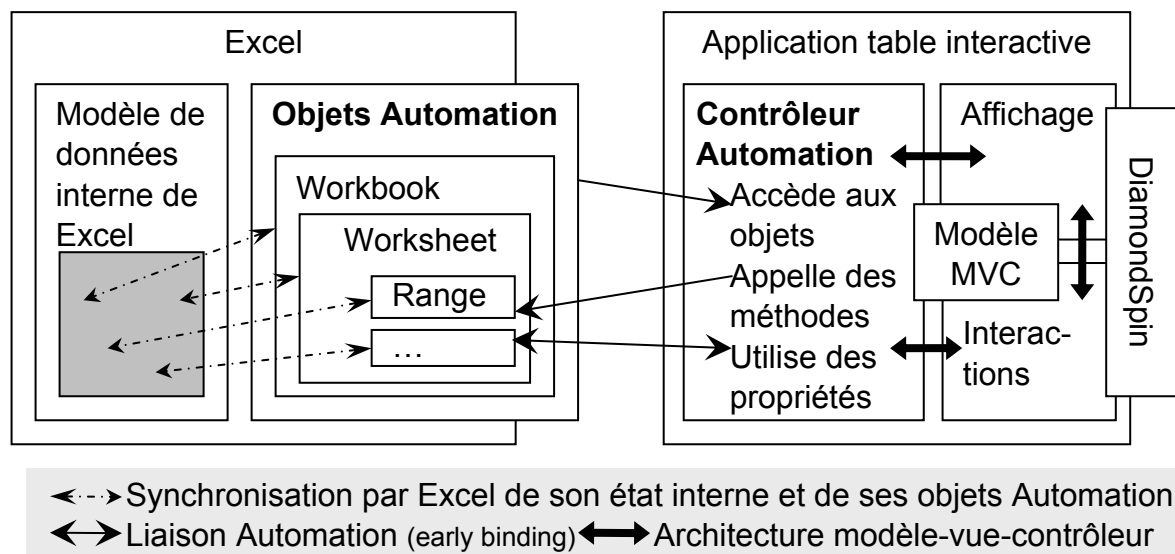


Figure 7 : Architecture du système basé sur la technologie de scripting Automation, permettant d'utiliser l'application existante Microsoft Excel avec une table interactive.

1.3.5. Accessibilité numérique

Les interfaces de programmation dédiées à l'accessibilité numérique sont utilisées pour offrir des interfaces alternatives à des groupes d'utilisateurs spécifiques, en particulier les utilisateurs souffrant de déficience visuelle [86, 117]. Elles permettent à un logiciel d'accessibilité d'obtenir des informations sur les applications en cours de fonctionnement sur un ordinateur de bureau, ainsi que sur leurs interfaces utilisateur et leurs états, pour présenter ces informations sous une forme adaptée à l'utilisateur, telle qu'une synthèse vocale. Ces technologies sont supportées par un grand nombre de systèmes d'exploitation et de boîtes à outils d'interface graphique tels que Microsoft Accessibility, des API Java, Mac OS X,... Le support de fonctionnalités

d'accessibilité numérique est maintenant requis par la loi dans plusieurs pays, et ne peut donc que s'améliorer.

Ces technologies permettent à des logiciels tiers d'obtenir des informations sémantiques à propos des interfaces utilisateur graphiques des applications en cours d'exécution. Les informations sont sémantiques pour permettre aux logiciels tiers de les « traduire » sous une autre forme que leurs représentations graphiques traditionnelles. Ces informations incluent pour chaque composant son nom, son rôle (bouton, menu, case à cocher,...), son texte et les autres éléments graphiques nécessaires pour le reproduire (bordure, couleur,...), son état (activé/désactivé, coché/décoché,...) et sa position à la fois sur l'écran en terme de pixel et dans la hiérarchie de l'interface utilisateur en terme de contenant et de contenu. Enfin, les actions que ce composant peut déclencher (par un clic, un survol avec le curseur de la souris, l'appui sur une ou plusieurs touches du clavier, ou d'autres événements d'interaction générés par un ordinateur de bureau) sont listées et peuvent être déclenchés programmatiquement.

Les technologies d'accessibilité numérique sont appropriées pour intégrer un module de commande vocale dans une application pour sélectionner vocalement parmi ses boutons ou ses menus, mais elles manquent de flexibilité pour opérer des changements plus profonds. En effet, ces technologies ont été conçues pour offrir un moyen d'accès alternatif à l'environnement traditionnel des ordinateurs de bureau, et non un nouvel environnement complet.

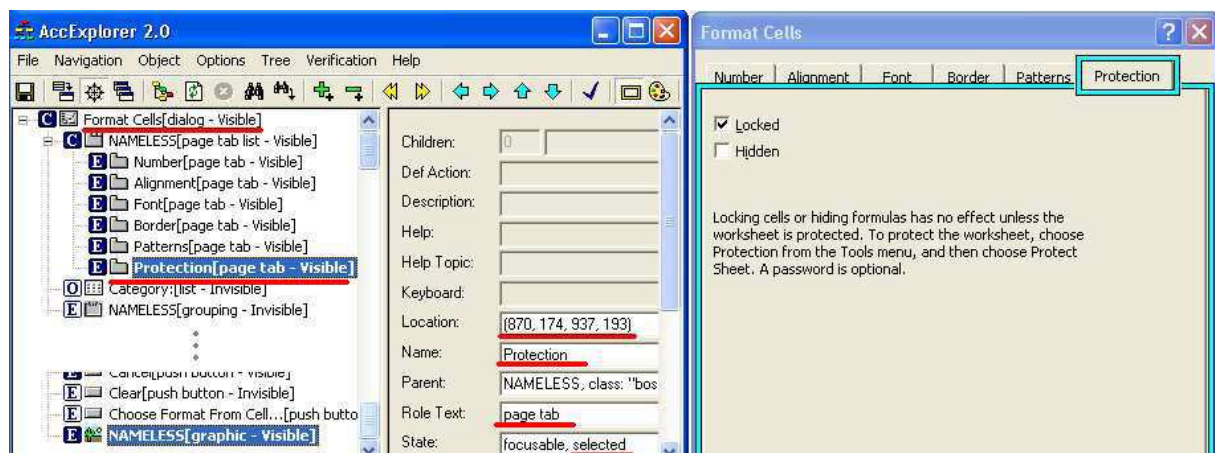


Figure 8 : Une boîte de dialogue de l'application existante Microsoft Excel (à droite), et une partie des informations d'accessibilité fournies par le logiciel *Accessibility Explorer tool* (de la boîte à outils Microsoft *Active Accessibility 2.0 SDK*) [9].

Plusieurs projets utilisent les informations fournies par l'accessibilité numérique (comme celles de la figure 8) pour segmenter des images précédemment obtenues via des captures d'écran. Berry *et al.* [9] les utilisent pour masquer certaines parties de l'interface utilisateur de Microsoft Excel. L'accessibilité numérique leur permet de travailler à un niveau de détails extrêmement fin, sur des cellules individuelles du

tableur. Le projet Façade [114] va plus loin en modifiant l'interface utilisateur d'origine selon un procédé que Stuerzlinger *et al.* appellent composition. Des parties de l'interface d'origine sont dupliquées et juxtaposées les unes à côté des autres pour créer une nouvelle interface. L'accessibilité numérique leur permet de déterminer la taille et la position de chaque composant. Un second mode permet également de changer le type du composant. Par exemple, une longue liste déroulante peut être changée en quelques cases à cocher pour les choix les plus fréquents, et un bouton pour accéder aux autres options. Ce second mode utilise les informations sur le contenu de la liste déroulante, qui sont impossible à obtenir par une simple capture d'écran.

1.3.6. Modification d'une boîte à outils d'interface homme machine

La dernière technologie consiste à réécrire des parties d'une boîte à outils d'interface homme machine. Virtuellement toutes les applications graphiques conçues depuis le milieu des années 80 utilisent une boîte à outils pour créer leur interface utilisateur, par exemple les boîtes à outils GTK+, Qt, Xaw ou Motif pour les systèmes d'exploitation basés sur X Window, Cocoa, Carbon ou Toolbox pour Mac OS, et win32 pour Microsoft Windows. Toutes ces boîtes à outils sont des couches d'abstraction logicielle entre l'application et les fonctionnalités d'entrée-sortie de l'ordinateur et de son système d'exploitation : elles gèrent une queue d'événements, et traduisent des mouvements de souris et des frappes clavier en commandes de haut niveau (par exemple, elles permettent à l'application de recevoir la commande de haut niveau « imprimer », et ce, que l'utilisateur ait cliqué sur le bouton « imprimer », l'ait sélectionné dans un menu, ou ait utilisé son raccourci clavier), elles fournissent une bibliothèque de composants d'interface utilisateur, chacun avec sa propre machine à états d'interaction (par exemple, quand l'utilisateur clique sur la barre de titre d'une fenêtre, l'application n'a pas besoin de gérer le déplacement de la souris, calculer la différence avec le point initial et bouger la fenêtre en fonction de cette différence : le composant barre de titre intègre ses comportements standards et les gère en interne) et ses fonctions de rendu graphique, et cetera. En réécrivant une partie d'une boîte à outils utilisée par une application, il est possible de sélectivement étendre ou remplacer un service ou un comportement donné de la boîte à outils pour le rendre compatible avec l'environnement de la table interactive.

Cette technologie peut atteindre des niveaux de structuration de donnée élevés, dans les deux sens donnés par la description de cet axe. En effet, en réécrivant les fonctions relatives à la création des composants de l'interface utilisateur, nous pouvons avoir une description structurée de cette interface et des relations entre les composants ; tandis qu'en réécrivant les fonctions relatives aux fonctionnalités courantes des interfaces utilisateur (telles que la gestion du presse-papier, les boîtes de dialogues standards ouvrir, sauvegarder, imprimer, la gestion du glisser déposer et des fonctionnalités d'échange de données, etc.) nous pouvons obtenir une description de certaines fonctionnalités de l'application et de ses processus internes.

Cette technologie a aussi un fort potentiel pour la flexibilité. Par exemple, elle permet d'avoir des boîtes de saisie de texte collaborative avec deux curseurs d'insertion en réécrivant les fonctions traitant les événements clavier et souris bruts et les convertissant en événements de haut niveau et les fonctions de rendu graphique des boîtes de saisie. Ou encore des cases à cocher qui s'activent par un mouvement de traversement au lieu d'un clic, en réécrivant les fonctions relatives à la machine des états interne des cases à cocher.

Les performances liées à cette technologie sont élevées car les communications entre l'application et le lien avec le système de table interactive utilisent le protocole prévu pour la communication entre l'application et sa boîte à outils d'interface homme machine. Il n'y a donc pas de conversion de données ou d'appel de code inutile. Il n'y a pas non plus besoin de réécrire l'application elle-même. Par contre, c'est la technologie la plus difficile à implémenter, avec plusieurs fonctions de la boîte à outils qui doivent être réécrites pour chaque changement de comportement que l'on veut mettre en œuvre.

Cette approche a été utilisée dès 1992 dans le projet Mercator [82] pour remplacer des fonctions de la boîte à outils de X Window par des fonctions d'une boîte à outils d'interface audio. En 1994, Chatty [18] modifie une boîte à outils, Whizz, pour permettre l'interaction bi manuelle, avec deux pointeurs de souris en parallèle. Toutefois, dans son approche, les applications doivent également être modifiées pour supporter les deux pointeurs.

Nous avons utilisé cette approche pour créer la boîte à outils DiamondSpin [108] pour le développement de nouvelles applications spécifiquement conçues pour les tables interactives. DiamondSpin se présente comme une extension de la boîte à outils Swing de Java : pour chaque classe standard (JFrame, JMenuBar, JComboBox,...), nous avons créé une nouvelle classe qui dérive la classe standard, via le mécanisme d'héritage de ce langage orienté objet, et ajoute ou redéfinit un certain nombre de ses méthodes. Les nouvelles méthodes implémentent les comportements spécifiques aux systèmes de tables interactives, tandis que les comportements communs aux ordinateurs de bureau et aux tables interactives sont hérités de la classe standard. Plusieurs composants internes à la boîte à outils, comme le mécanisme de gestion des événements ou le système de sortie graphique, ont été réécrits complètement. Ainsi, DiamondSpin gère les événements dans plusieurs threads concurrents pour permettre à plusieurs utilisateurs d'interagir en même temps avec la table, en parallèle, sans risquer de se bloquer les uns les autres par une action modale par exemple ; tandis que la sortie graphique se fait grâce à l'API graphique OpenGL qui permet à des effets comme la rotation des fenêtres d'être produits en temps réel.

Une approche similaire utilisant également la dérivation et l'héritage des langages orientés objets a été utilisée par la boîte à outils *subArctic* [49]. Les concepteurs de *subArctic* utilisent les mécanismes intégrés dans le langage de programmation sous jacent, pour étendre des objets graphiques. Le programmeur de l'application n'a pas à connaître ces extensions pour écrire son application : elles sont automatiquement activées quand un objet graphique les requiert [31].

Nous avons utilisé DiamondSpin pour créer des dizaines d'applications, et noté des tendances dans notre façon de les programmer : nous n'utilisons pratiquement jamais les nouvelles méthodes ajoutées par DiamondSpin directement. Nous utilisons le plus souvent les méthodes Swing existantes, en sachant que leur comportement a été redéfini pour correspondre au comportement attendu d'une application pour tables interactives. Par exemple, nous utilisons la méthode `setFocus` pour amener une fenêtre au premier plan, en sachant que cela ne va pas supprimer le focus des fenêtres des autres utilisateurs de la table interactive. Les quelques méthodes entièrement nouvelles dans DiamondSpin sont utilisées principalement pour initialiser le matériel et configurer l'environnement de l'utilisateur (politique de partage de documents, d'orientation des fenêtres, et d'autres fonctionnalités spécifiques aux systèmes de tables interactives). Dans cette approche, la boîte à outils DiamondSpin est une évolution de la boîte à outils Swing, et non une nouvelle boîte à outils, ce qui lui permet d'être compatible avec certaines applications existantes (au renommage des classes près), bien que ce n'était pas notre but initial lors de la création de DiamondSpin.

Nous avons étendu cette approche à une boîte à outils plus courante que Swing avec le but principal d'utiliser des applications existantes dans un environnement de table interactive, sans modifier les applications en question. Nous avons également cherché à réellement intégrer les applications dans le modèle d'interaction de la table interactive, en prenant en compte les travaux existants dans le domaine des techniques d'interaction et de la visualisation d'information adaptés aux tables interactives. Pour implémenter cette extension de boîte à outils d'interface utilisateur, nous présentons d'abord notre démarche pour choisir quelle boîte à outils modifier. En effet, les différentes boîtes à outils présentent une interface de programmation (API) différente, ce qui nous impose de faire le choix d'une seule boîte à outils. Nous devons également lier les applications existantes, non modifiées, avec la nouvelle boîte à outils. Enfin, les boîtes à outils d'interface utilisateur intègrent des milliers de fonctions, et nous ne proposons bien entendu pas de les réécrire toutes. Nous distinguons trois niveaux de fonctionnalités qui peuvent être atteints en modifiant différentes sous parties de la boîte à outils. Nous avons également exploré l'interdépendance entre les fonctions existantes de la boîte à outils pour minimiser les effets de bords de nos modifications.

Les boîtes à outils d'interface homme machine sont des couches d'abstraction logicielles entre les applications d'une part et le système d'exploitation et les périphériques d'autre part. Ces boîtes à outils gèrent une queue d'événements, traduisent les commandes clavier et souris en commandes de haut niveau, fournissent une bibliothèque de composants d'interface utilisateur, et plus encore. Malgré ces fonctionnalités communes, les différentes boîtes à outils diffèrent grandement dans leur implémentation. Certaines boîtes à outils sont orientées objet, comme Cocoa pour Mac OS, ou Qt et Xaw pour X Window, tandis que d'autres sont procédurales, comme Carbon et Toolbox pour Mac OS, win32 pour Microsoft Windows, ou GTK+ pour X Window. Certaines boîtes à outils sont natives, intégrées dans leur système d'exploitation (Carbon, Toolbox et win32 sont des boîtes à outils natives de leurs systèmes d'exploitation respectifs, et il n'y a pas de boîte à outils

native pour les systèmes basés sur X Window), tandis que d'autres sont des couches séparées, parfois de plus haut niveau, basées sur les boîtes à outils natives ou de plus bas niveau. Des dizaines de boîtes à outils existent dans cette dernière catégorie, on peut citer Cocoa, MacApp et MacZoop pour Mac OS, et MFC, WTL, .NET, WPF pour Microsoft Windows, toutes basées sur la boîte à outils procédurale win32. Certaines boîtes à outils de haut niveau peuvent fonctionner avec plusieurs boîtes à outils de bas niveau, ce qui leur permet d'être multi-plateforme : ainsi une application conçue avec la boîte à outils Qt peut utiliser indirectement les boîtes à outils natives de Mac OS, Microsoft Windows et fonctionner avec X Window. L'application elle-même ne voit que l'interface commune, unifiée de Qt, qui abstrait les différences d'implémentation entre les boîtes à outils de plus bas niveau.

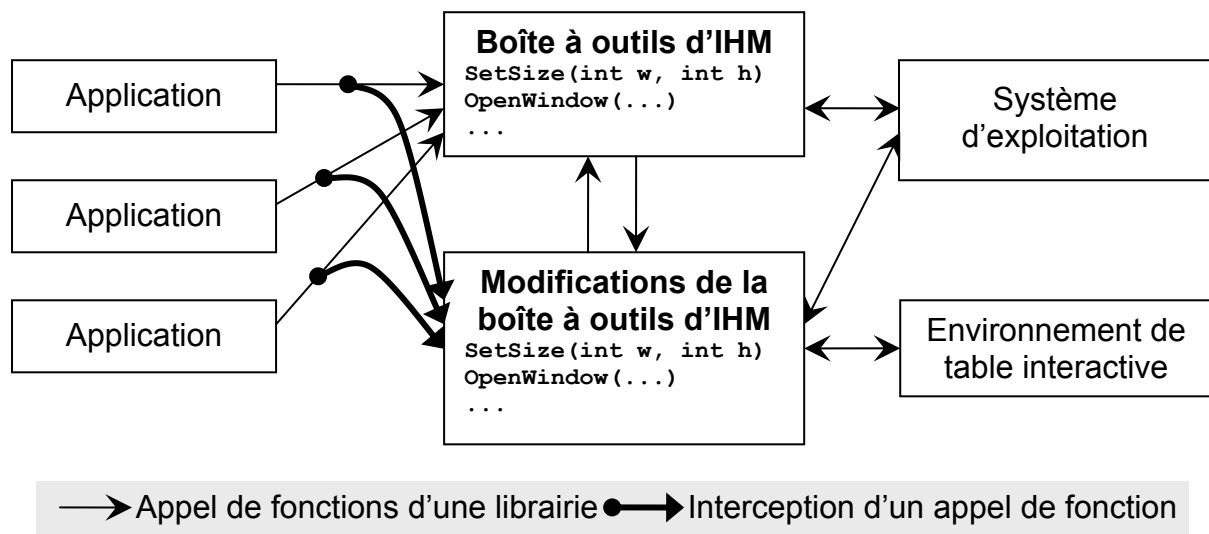


Figure 9 : Architecture du système utilisant des modifications d'une boîte à outils d'interface utilisateur existante, pour permettre l'utilisation d'applications existantes non modifiées sur une table interactive.

Pour atteindre notre objectif d'utiliser des applications existantes dans un environnement de table interactive via l'extension d'une boîte à outils, nous devons donc choisir quelle boîte à outils partiellement réécrire. Nous pensons que le point le plus important est le nombre d'applications propriétaires utilisant la boîte à outils. En effet, d'une part les applications libres sont plus multi-plateforme [89] que les applications propriétaires, statistiquement il y aura donc plus d'applications libres que d'applications propriétaires conçues pour fonctionner directement avec un environnement de table interactive, sans avoir besoin d'une couche d'adaptation entre l'application et la table interactive. D'autre part, les applications propriétaires, y compris les applications existantes, qui ne sont pas nativement compatibles avec les tables interactives ne peuvent pas être modifiées pour le devenir. Au contraire, les applications libres, puisque leur code source est public, peuvent être modifiées individuellement pour interagir avec un système de table interactive. Pour les applications propriétaires, seul un composant externe assurant le lien entre l'application et l'environnement de table interactive peut permettre cette compatibilité, d'où l'intérêt de maximiser le nombre d'applications propriétaires

concernées par la réécriture partielle de leur boîte à outils d'interface homme machine.

1.3.6.1. Choix d'une boîte à outils

Le second point à considérer pour le choix de la boîte à outils à modifier est la difficulté d'implémenter de telles modifications. Chaque fonction de la boîte à outils devrait être soigneusement documentée, pour permettre de minimiser au maximum les conséquences et les effets de bord de la modification d'une fonction. Une boîte à outils libre, avec son code source rendu public, permet de savoir plus précisément quelles fonctions influent sur quelles autres fonctions.

Nous avons choisi d'étendre la boîte à outils win32, la boîte à outils native de Microsoft Windows. En effet, toutes les applications graphiques fonctionnant sous Microsoft Windows utilisent cette boîte à outils, soit directement, soit indirectement via une boîte à outils de plus haut niveau. D'autre part, selon notre étude informelle des besoins des utilisateurs, les applications les plus demandées fonctionnent quasiment toutes sous Microsoft Windows. Du point de vue de la difficulté d'implémentation, toutes les fonctions publiques de l'interface de programmation de win32 sont documentées (et Microsoft devrait révéler la documentation de ses interfaces et protocoles privés, selon des décisions de la Commission européenne). Un sous-ensemble de ceux-ci a été normalisé par l'*European Computer Manufacturers Association* dans la norme ECMA-234 [36]. Nous disposons également de plusieurs implémentations libres de cette interface de programmation : les projets Wine [128] et PEACE [90]. Ces projets ont pour but d'écrire une implémentation entièrement nouvelle de l'interface de programmation win32, dans le but de faire fonctionner des programmes conçus pour Microsoft Windows sur d'autres systèmes d'exploitation. En revanche, ils conservent l'environnement d'ordinateur de bureau auquel ces programmes s'attendent. Pour notre part, notre but est de faire fonctionner des applications conçues pour des ordinateurs de bureau dans un environnement différent : une table interactive, nous n'avons donc pas besoin de réécrire l'intégralité de l'interface de programmation, mais par contre il convient d'en modifier une partie. Ces projets sont tout de même utiles : leur code source permet de comprendre le fonctionnement interne de certaines fonctions. Une évaluation de cette utilité se trouve détaillée ci-après.

1.3.6.2. Lien entre la nouvelle boîte à outils et les applications existantes

Maintenant que nous avons choisi une boîte à outils, nous devons lier les applications existantes, non modifiées, avec la nouvelle boîte à outils. En effet, les fonctions que nous nous proposons de modifier ne font pas partie de l'application elle-même : elles sont dans un ensemble de bibliothèques de fonctions. Quand elles démarrent, les applications recherchent ces bibliothèques et se lient dynamiquement avec les fonctions de ces dernières dont elles ont besoin [45]. Nous avons plusieurs options pour influencer sur ce mécanisme de liaison.

Si nous disposons du code source de l'application et de la bibliothèque de fonctions, nous pouvons simplement modifier le code source de la bibliothèque selon nos

besoin, et statiquement re-lie l'application (sans la modifier) avec la nouvelle bibliothèque. Notons que dans notre cas précis, les bibliothèques win32 ne sont pas libres, mais nous pouvons utiliser les implémentations Wine ou PEACE comme base.

Si nous disposons juste du code source de l'application, nous pouvons écrire une bibliothèque contenant juste les fonctions que nous avons réécrites, et de simples fonctions proxy pour les autres. Nous lions notre bibliothèque avec la bibliothèque originale (les fonctions proxy sont de simples fonctions d'une seule ligne qui appellent les fonctions de la bibliothèque non modifiées), et l'application avec notre bibliothèque.

Si nous ne disposons ni du code source de l'application, ni de celui de la boîte à outils d'interface utilisateur qu'elle utilise, ce qui est le cas pour la majorité des applications identifiées par notre étude informelle des besoins des utilisateurs, nous devons alors intercepter dynamiquement les appels aux fonctions de la bibliothèque et rerouter ces appels vers nos propres fonctions. Cette approche est utilisée par exemple par le Center for Bioinformatics and Computational Biology de l'Université du Maryland [81]. Un grand nombre d'applications existantes de bioinformatique ont été conçues pour l'environnement traditionnel d'un serveur de calcul. Les auteurs de ces travaux veulent utiliser ces applications existantes avec une plateforme de calcul distribué : BOINC (SETI@Home, the World Computing Grid,...). Les plateformes de calcul distribué demandent une programmation des applications différente de celle des serveurs de calcul. Plutôt que de réécrire chacune de ces applications, ou même d'en recompiler les codes sources, ils ont écrit une couche d'adaptation entre l'application et la plateforme (dans leur cas, il s'agissait de réécrire les fonctions standards de gestion des fichiers, cette dernière étant prise en charge par BOINC) pour dynamiquement intercepter les appels à ces fonctions. Les applications de l'Université du Maryland comprennent des logiciels pour Microsoft Windows, Mac OS et Unix, ils ont donc exploré différentes possibilités d'interception de fonctions binaires sur ces trois systèmes d'exploitation. Pour notre part, nous n'utiliserons que la version pour Microsoft Windows : le système Detours de Microsoft Research [50].

Le projet Chromium utilise également une approche similaire : il intercepte les appels aux fonctions de rendus 3D de OpenGL et modifie leur comportement à la volée, par exemple pour ajouter des plans de coupe afin de produire des vues « explosées » d'environnement architecturaux en 3D [84].

Afin de fonctionner, les fonctions de la couche d'adaptation doivent avoir exactement la même signature (arguments et type de retour) que les fonctions d'origine qu'elles remplacent. En effet, elles sont appelées à la place de ces fonctions, et la gestion de pile d'appel est à la charge de l'application, non modifiée, appelante. En revanche, le concepteur peut exécuter n'importe quel traitements nécessaires dans les fonctions de la couche d'adaptation : par exemple appeler d'autres fonctions (e.g. les fonctions de BOINC), appeler au moment opportun la fonction d'origine qu'elle remplace, avec ou sans modification des arguments, ou exécuter des algorithmes de son choix.

1.3.6.3. Types de modifications

Nous avons choisi de modifier une boîte à outils spécifique, la boîte à outils win32, et nous avons une technologie pour lier les applications existantes à nos modifications. Toutefois les boîtes à outils d'interface utilisateur regroupent des milliers de fonctions, et nous ne proposons bien entendu pas de les réécrire toutes. Un tel projet sort de faisabilité d'une thèse de doctorat. Nous voyons trois niveaux de fonctionnalités qui peuvent être atteints en modifiant différentes sous parties de la boîte à outils d'interface homme-machine.

Le premier niveau de fonctionnalité consiste à obtenir des informations sur l'application, de façon similaire à la technologie de l'interface d'accessibilité. En effet, l'application utilise les fonctions de la boîte à outils d'interface utilisateur pour créer chaque composant de cette interface un par un, et définir leurs relations par rapport aux autres, ainsi que pour mettre à jour cette interface. Dans cet objectif, des fonctions « proxy » très simples, qui appellent les fonctions win32 d'origine après avoir examiné les paramètres et/ou les valeurs de retour, suffisent.

Le second niveau de fonctionnalité consiste à ajouter des comportements ou des interactions spécifiques aux tables interactives aux applications. Par exemple, permettre la rotation des fenêtres ou le déplacement des menus surgissants. Ce type de modification nécessite d'associer des données additionnelles aux éléments existants. Comme dans le cas précédant, les fonctions modifiées appellent les fonctions d'origine pour tous les traitements concernant les données « traditionnelles », déjà présentes dans la boîte à outils, associées à un composant. En revanche, les fonctions modifiées doivent aussi traiter les données additionnelles de façon cohérente. Par exemple, pour permettre la rotation des fenêtres, il faut associer à chaque fenêtre un angle de rotation et un point autour duquel la rotation se fait. La fonction qui affiche la fenêtre doit être modifiée pour appliquer une transformation (matrice de rotation) adéquate au contexte graphique, avant d'appeler la fonction d'affichage d'origine. Cette dernière utilisera les informations stockées par la boîte à outils, telle que la position de la fenêtre. Pour le second exemple, le déplacement des menus surgissants nécessite d'ajouter des interactions pour déplacer le menu, par exemple une poignée de déplacement, et donc de modifier la fonction qui gère les événements du menu pour prendre en compte ces nouvelles interactions. La modification de la position du menu, que la boîte à outils considère comme une constante une fois que le menu a été affiché, nécessite également de changer la fonction d'affichage pour la prendre en compte.

Enfin, le troisième niveau de fonctionnalité consiste à modifier ou supprimer des comportements ou des interactions implémentés par la boîte à outils. Par exemple, activer des boutons par un geste de franchissement plutôt que par un clic de souris, ou encore permettre à plusieurs utilisateurs d'avoir chacun un focus clavier distinct. Ces buts nous semblent parfois incompatibles avec certaines structures de données utilisées par la boîte à outils : par exemple la boîte à outils utilise une unique variable pour stocker l'identifiant du composant d'interface actuellement actif, et il est impossible de stocker plus d'un identifiant dans l'espace de cette variable. Il convient

donc de créer des structures de données propres, ce qui implique de réécrire toutes les fonctions qui créent, détruisent ou accèdent aux instances des structures en question. Contrairement aux deux niveaux de fonctionnalité précédents, ces nouvelles fonctions ne vont pas appeler les fonctions d'origine de la boîte à outils, et les structures de données d'origine ne seront en conséquence jamais instanciées ni utilisées. C'est le type de modification le plus complexe, car doivent être réécrites toutes les fonctions qui utilisent les structures de données concernées. Afin de mesurer plus exactement cette difficulté, nous avons mené une expérience destinée à vérifier le niveau d'interdépendance et de redondance dans l'implémentation de Microsoft de la boîte à outils win32. Nous avons également confirmé ces résultats en examinant le code des implémentations des projets Wine [128] et PEACE [90], qui visent à créer des implémentations indépendantes de la boîte à outils win32.

Nous avons identifié un ensemble de fonctions d'interface utilisateur qui soit aussi indépendant que possible du reste des autres fonctions, et mesuré l'interdépendance et la redondance entre ces fonctions. Il s'agit des fonctions relatives à l'API des menus (barre de menus et menus surgissants). Nous avons écrit une librairie de menus qui remplace chacune des fonctions d'origine de l'API de menus par une fonction qui affiche simplement son nom, ses arguments et appelle la fonction d'origine. Les fonctions de remplacement ont exactement les mêmes arguments et les mêmes types de retour que les fonctions d'origine. En utilisant plusieurs applications, couvrant les divers cas d'utilisation possible des menus, avec cette librairie de remplacement, nous avons trouvé que la majorité des fonctions de l'API de menus de win32 sont en fait des raccourcis ou des pré-processeurs pour quelques fonctions centrales de l'API. Seules ces fonctions centrales accèdent directement aux structures de données internes. Les fonctions raccourcis ou pré-processeurs sont indépendantes des structures de données internes et n'ont donc pas besoin d'être réécrites si celles-ci changent.

Ainsi, par exemple, la fonction *LoadMenu(ResourceID)* appelle les fonctions *FindResource* et *LoadResource* avec l'identifiant de ressource *ResourceID* pour charger la description du menu en mémoire. Ensuite, elle appelle *LoadMenuIndirect(MenuDescriptor)* sur le descripteur du menu obtenu précédemment. *LoadMenuIndirect* analyse le descripteur du menu et appelle la fonction *CreateMenu()* pour créer un menu vide, puis plusieurs fois la fonction *InsertMenuItem(Menu, ItemName,...)* pour créer les éléments du menus (avec les paramètres appropriés, tels qu'extraits du descripteur). *LoadMenuIndirect* s'appelle également récursivement si le descripteur du menu inclut un sous-menu. Si nous voulons intercepter la création des menus d'une application, par exemple pour les remplacer par des menus circulaires, nous n'aurions qu'à réécrire *CreateMenu* et *InsertMenuItem* : quelle que soit la façon dont l'application crée ses menus (à partir d'une ressource, d'un descripteur de menu obtenu d'une autre source, ou dynamiquement en appelant les fonctions *CreateMenu* et *InsertMenuItem*), ces deux dernières fonctions sont les seules qui créent effectivement des instances de la structure de données des menus. En particulier, nous n'avons pas besoin de comprendre la syntaxe des descripteurs de menus ni de modifier le code source d'analyse de ce descripteur dans *LoadMenuIndirect*.

Nous avons également utilisé les implémentations libres de l'API win32 produites par les projets Wine et PEACE. Elles confirment l'arbre d'appels des fonctions que nous avons observé avec la bibliothèque de menus de remplacement. Par exemple, le code source suivant pour la fonction *LoadMenu* provient de l'implémentation Wine :

```
HMENU WINAPI LoadMenu(HINSTANCE instance, LPCSTR name){
    HRSRC hrsrc = FindResource(instance, name, (LPSTR)RT_MENU);
    if (!hrsrc) return 0;
    return LoadMenuIndirect(
        (LPCVOID)LoadResource(instance, hrsrc));
}
```

Celui-ci est la même fonction dans l'implémentation PEACE :

```
HMENU WINAPI LoadMenu(HINSTANCE h, LPCWSTR name){
    HRSRC r;
    HGLOBAL rh;
    MENUTEMPLATE *tmpl;
    if ((r = FindResource(h, name, (LPCWSTR)RT_MENU)) == 0)
        return 0;
    if ((rh = LoadResource(h, r)) == 0)
        return 0;
    tmpl = LockResource(rh);
    return LoadMenuIndirect(tmpl);
}
```

Ces deux implémentations confirment à la fois l'utilité des ces sources comme outils pour comprendre le fonctionnement interne de l'interface de programmation d'application (API) win32, et l'indépendance entre de nombreuses fonctions et les structures de données utilisées (ici, *MENUTEMPLATE* et *HMENU*). Si nous changeons ces structures de données pour stocker les informations supplémentaires requises pour gérer l'interaction sur une table interactive, par exemple pour ajouter l'angle de rotation du menu, une partie des fonctions existantes de l'API win32 n'aura pas besoin d'être réécrite. Elles utiliseront les nouvelles structures de données au travers des pointeurs opaques *HMENU*¹ et *MENUTEMPLATE**, car ils ne sont pas déréférencés dans ces fonctions. Notre étude des arbres d'appels des fonctions concernant les menus nous indique que sur un total de 39 fonctions utilisant les types de données concernés, la moitié (20 fonctions) accèdent directement aux données stockées dans ces types. Nous appelons ces fonctions, des fonctions noyaux (*core functions*). Parmi ces 20 fonctions noyaux, 5 sont des accesseurs (*getters*) triviaux : elles accèdent directement à un champ du type de données et retourne son contenu à une fonction plus complexe. Ainsi la fonction plus complexe ne dépend pas de la représentation des données en mémoire. Enfin, les 19 fonctions restantes, qui ne sont pas des fonctions noyaux, n'accèdent pas directement aux données en mémoire. Elles

¹ Le H (pour *handle*) de *HMENU* indique qu'il s'agit d'un pointeur respectant les conventions win32. L'astérisque de *MENUTEMPLATE** indique qu'il s'agit d'un pointeur C standard.

utilisent les fonctions noyaux, entre autres les accesseurs, pour cela. La liste des 39 fonctions de menus et leur caractérisation est fournie à la figure 10.

Nom de la fonction	Fonction noyau ?	Détails
CreateMenu DestroyMenu InsertMenuItem	Oui	Gèrent la mémoire utilisée par les structures de données. Initialisent les structures de données.
GetMenuItemCount GetMenuItemID GetMenuState GetMenuString GetSubMenu	Oui	Accesseurs triviaux.
GetMenuInfo GetMenuBarInfo GetMenuItemInfo	Oui	Convertissent les données contenues dans les structures de données internes (dont le format n'est pas documenté) en un format documenté en lecture seule.
CheckMenuItem CheckMenuRadioItem GetMenuDefaultItem HiliteMenuItem ModifyMenu RemoveMenu SetMenuDefaultItem	Oui	Implémentent des algorithmes qui accèdent directement aux structures de données des menus.
EnableMenuItem	Oui	Utilise plusieurs fonctions relatives aux menus, mais accède également directement aux structures de données des menus.
DrawMenuBar	Oui	Accède également directement aux structures de données des menus et appelle plusieurs fonctions externes de dessins pour afficher les menus à l'écran.
LoadMenuIndirect	Non	Appelle <i>CreateMenu</i> , <i>InsertMenuItem</i> , et <i>LoadMenuIndirect</i> (récursive). Cf. exemple ci-dessus.
LoadMenu	Non	Appelle <i>LoadMenuIndirect</i> . Cf. exemple ci-dessus.
CreatePopupMenu	Non	Appelle simplement <i>CreateMenu</i> avec le drapeau <i>MF_POPUP</i> .
DeleteMenu	Non	Appelle <i>DestroyMenu</i> et <i>RemoveMenu</i> .
InsertMenu	Non	Appelle <i>InsertMenuItem</i> avec des valeurs par défaut pour les paramètres additionnels.
AppendMenu	Non	Appelle <i>InsertMenu</i> avec le paramètre <i>position</i> à -1

TrackPopupMenu TrackPopupMenuEx EndMenu	Non	Appellent des fonctions relatives à la gestion de la queue d'événements avec des valeurs par défaut pour certains paramètres.
GetMenu	Non	Appelle des fonctions génériques de win32 pour gérer les structures de données opaques de type <i>handle</i> .
SetMenu	Non	Appelle <i>IsMenu</i> et des fonctions génériques de win32 pour gérer les structures de données opaques de type <i>handle</i> .
IsMenu	Non	Appelle <i>GetMenu</i> , et retourne proprement une erreur si <i>GetMenu</i> a échoué.
GetSystemMenu	Non	Utilise <i>GetMenu</i> pour accéder à un menu particulier.
GetMenuCheckMark- Dimensions GetMenuItemRect	Non	Utilisent des fonctions génériques de win32 pour convertir des coordonnées écran.
MenuItemFromPoint	Non	Utilise <i>GetMenuItemRect</i> pour comparer des coordonnées.
SetMenuInfo SetMenuItemBitmaps SetMenuItemInfo	Non	Calculent des modifications et appellent <i>ModifyMenu</i> pour les stocker.

Figure 10 : Les fonctions de l'interface de programmation d'application (API) win32 concernant aux menus, et la façon dont elles interagissent entre elles et avec les structures de données de menus.

1.3.6.4. Implémentation

Pour valider cette approche, nous avons développé et implémenté deux preuves de concept utilisant cette technologie. La première modifie le comportement des menus déroulants et surgissants d'une application. Le comportement par défaut des menus sous Windows est supprimé et remplacé par des menus personnalisés. La seconde preuve de concept utilise notre librairie d'interface utilisateur pour table interactive DiamondSpin [108] associée aux contenus de fenêtres d'applications existantes.

Pour la première preuve de concept de la technologie de réécriture des parties d'une boîte à outils d'interface homme machine, nous avons remplacé les fonctions de menus précédemment identifiées comme étant des fonctions noyaux, c'est-à-dire qui accèdent directement au contenu des structures de données de menus (détaillées dans la figure 10). Ces modifications correspondent au troisième niveau de fonctionnalité précédemment identifié. Les fonctions que nous avons écrites pour remplacer les fonctions noyaux du système de menu n'appellent pas ces fonctions d'origine. A la place, nos fonctions maintiennent leurs propres structures de données. Comme nos fonctions n'appellent pas les fonctions d'origine qu'elles remplacent, la boîte à outils win32 n'a pas l'opportunité de créer les « vrais » menus, au sens de ceux que l'application qui a appelé ces fonctions s'attendait à créer. En conséquence, nous avons donc aussi dû réécrire les fonctions qui accèdent aux données contenues dans

nos structures de données. L'ensemble de ces fonctions a été compilé dans une librairie dynamique (DLL) et le système Detours [50] a été utilisé pour pouvoir insérer cette librairie dans n'importe quelle application existante sous Windows.

La librairie produite pour cette première preuve de concept n'est pas spécifique à l'environnement des tables interactives. Elle permet d'utiliser les menus d'une application depuis un autre ordinateur. Les menus existants sont graphiquement supprimés des applications concernées, et affichés dans une interface centralisée sur un autre ordinateur via un lien réseau. L'interface peut être utilisée pour sélectionner des éléments dans les menus affichés, des événements sont alors transmis en sens inverse vers les applications. Les applications ne sont pas modifiées, et traitent ces événements comme des événements locaux, générés par des menus traditionnels, seraient traités.

Les principaux points intéressants de cette preuve de concept sont les fonctions *DrawMenuBar*, *CreateMenu*, *DestroyMenu*, *InsertMenuItem*, et *SetMenu*. Nous les détaillons une par une dans les paragraphes qui suivent.

La fonction *DrawMenuBar* est appelée par la boîte à outils d'interface utilisateur quand la barre de menu d'une fenêtre a besoin d'être dessinée. Par exemple, quand la fenêtre est affichée pour la première fois, ou quand une partie de la fenêtre qui était précédemment masquée par d'autres éléments de l'interface utilisateur est maintenant visible. Notre implémentation de cette fonction ne fait tout simplement rien, puisque les menus sont affichés sur une interface centralisée sur un autre ordinateur.

CreateMenu, *DestroyMenu*, et *InsertMenuItem* sont utilisés, directement ou indirectement comme nous l'avons vu précédemment, pour créer et manipuler le contenu des menus. Notre implémentation utilise les paramètres passés à ces fonctions par les applications existantes pour mettre à jour nos propres structures de données. Ces modifications sont propagées à l'interface centrale qui affiche les menus, mais pas à la boîte à outils d'interface utilisateur. Du point de vue de celle-ci, l'application n'a pas demandé la création de menus. En revanche, du point de vue de l'application, ses appels aux fonctions de création de menus ont été accomplis avec succès. Notre librairie doit donc aussi fournir les fonctions accesseurs que l'application peut utiliser pour obtenir des informations sur les menus qu'elle a créés. En aucun cas notre librairie ne peut permettre à une application de communiquer directement avec la boîte à outils d'interface utilisateur au sujet de ses menus, puisque les informations dont ces deux entités disposent sur l'existence de menus sont contradictoires.

Enfin, la fonction *SetMenu* est utilisée par les applications pour associer un menu précédemment créé avec une fenêtre précédemment créée. Comme le menu n'a pas été créé du point de vue de l'interface de programmation d'application (API) win32, notre implémentation de cette fonction n'appelle pas la fonction d'origine de win32. A la place, elle stocke l'identifiant de la fenêtre dans la structure de données

contenant la description du menu. Cet identifiant est ensuite utilisé pour envoyer les événements de sélection d'éléments du menu à la fenêtre concernée.

La librairie de détournement des menus Windows contient un total de 21 fonctions (les 20 fonctions noyaux et la fonction *SetMenu*), et l'implémentation de ces fonctions utilise 500 lignes de code en langage C. L'interface centrale d'affichage des menus, qui communique via le réseau TCP/IP avec la librairie, est implémentée en 200 lignes de code de Java (en mode console). Nous avons détourné les menus des applications intégrées à Windows (Bloc-notes, Paint, Calculatrice) et d'applications tierces parties (VLC, Acrobat Reader, Notepad++) avec succès, démontrant la faisabilité de cette approche.

La seconde preuve de concept que nous avons implémentée permet d'afficher le contenu de fenêtres d'applications existantes sous Windows dans l'environnement de tables interactives DiamondSpin [108]. DiamondSpin permet de programmer des applications pour tables interactives en utilisant le langage de programmation Java et ses composants d'interface utilisateur existants (librairie Swing [29]). DiamondSpin propose en plus des techniques d'interaction spécifiques aux tables interactives comme la rotation ou le pliage de fenêtres, des barres de menus coulissantes sur le bord de la surface d'affichage ou le drag-and-drop d'éléments d'un utilisateur à l'autre.

Comme dans l'implémentation précédente, nous avons implémenté une librairie contenant des fonctions de substitution, qui vont venir remplacer des fonctions existantes de la boîte à outils d'interface utilisateur win32. Cependant, à la différence de l'implémentation précédente, chacune de nos fonctions de remplacement appelle la fonction win32 d'origine avant de retourner le contrôle à l'application. Cette librairie met en œuvre une stratégie du type du deuxième niveau de fonctionnalités mis en évidence précédemment. Par exemple, notre librairie remplace la fonction win32 *CreateWindow* qui est appelée par l'application pour créer des fenêtres². A chaque fois qu'une application appelle la fonction *CreateWindow*, notre librairie appelle la fonction équivalente de DiamondSpin, c'est-à-dire le constructeur de la classe *DSFrame*, pour créer une fenêtre sur la table interactive. Le titre, la taille et le style de cette fenêtre sont définis en traduisant les paramètres fournis à *CreateWindow* en paramètres compatibles avec la fonction de DiamondSpin. Ensuite, notre librairie appelle la fonction *CreateWindow* d'origine de win32. Celle-ci crée une fenêtre win32, identifiée par un *handle* (pointeur opaque spécifique à win32) de type *HWND*. Notre fonction enregistre le lien entre ces deux fenêtres, la fenêtre créée par DiamondSpin et celle créée par win32, dans une table de hachage (figure 11). De manière similaire, notre librairie intercepte les appels à la fonction *DestroyWindow*, et décrémente le compteur d'utilisation de l'instance de *DSFrame*

² La fonction *CreateWindow* est également utilisée, dans l'API win32, pour créer des fenêtres spéciales, comme des fenêtres « *message-only* », des fenêtres d'échanges DDE (*Dynamic Data Exchange*), et d'autres types de fenêtre non interactives et non visibles. Dans ces cas là, notre fonction de remplacement appelle simplement la fonction *CreateWindow* d'origine sans plus de traitements.

Données de l'API win32

Données de DiamondSpin

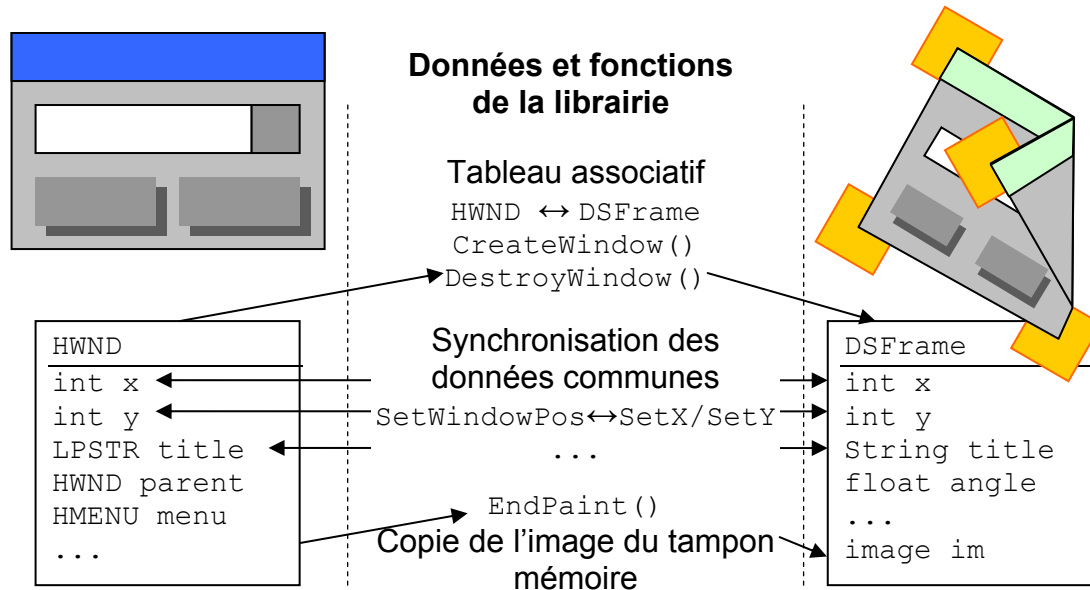


Figure 11 : Architecture de notre deuxième preuve de concept. Une librairie intermédiaire fait le lien entre les données de l'interface de programmation d'application (API) win32 et les données de la boîte à outils pour table interactive DiamondSpin.

correspondante à la fenêtre win32 détruite (DiamondSpin étant écrit en Java, l'instance sera détruite par le ramasse miette de Java).

Notre librairie est liée à deux boîtes à outils d'interface utilisateur, win32 et DiamondSpin, qui ont chacune été développée indépendamment dans le but de gérer l'ensemble de l'interaction avec le ou les utilisateurs. En conséquence, chacune stocke dans ses structures de données les informations de bases sur les fenêtres telles que le titre, la position, la taille, etc. Lors de la création d'une fenêtre, les données de DiamondSpin sont initialisées avec les valeurs des paramètres de la fonction *CreateWindow*, qui servent aussi à initialiser les données de win32. Mais ensuite, lors de l'exécution de l'application, ces données stockées de manière redondante par chaque boîte à outils peuvent se désynchroniser : l'utilisateur peut déplacer une fenêtre DiamondSpin sur la table interactive, ou l'application peut changer le titre de sa fenêtre win32. Certains paramètres sont successibles d'être modifiés par les deux parties : l'utilisateur peut redimensionner une fenêtre DiamondSpin alors que l'application peut adapter la taille de sa fenêtre win32 aux données qu'elle contient.

Pour conserver ces données redondantes synchronisées, notre librairie intercepte les fonctions win32 que les applications peuvent appeler pour les modifier. Lorsqu'une de ces fonctions est appelée, notre librairie appelle la fonction équivalente de DiamondSpin, puis propage l'appel à la fonction win32 d'origine. De façon similaire, les modifications initiées par un utilisateur, c'est-à-dire via DiamondSpin, sont capturées en utilisant le mécanisme d'écouteurs (*listener*) existant de Swing et DiamondSpin. Elles sont propagées aux structures de données de win32 grâce aux

mêmes fonctions que les applications utilisent pour initier ces changements (voir figure 11).

Les attributs non partagés, qu'ils fassent partie des données de *DiamondSpin* (e.g. l'angle de rotation de la fenêtre) ou de *win32* (e.g. l'identifiant de la fenêtre parente), sont pris en charge par la boîte à outils concernée en utilisant les mécanismes existants.

Dans cette implémentation de cette preuve de concept, le contenu des fenêtres est géré comme une image bitmap. Cette image bitmap est obtenue en modifiant deux fonctions de la boîte à outils *win32* : *BeginPaint* et *EndPaint*. *BeginPaint* prépare une fenêtre à être redessinée, et indique à l'application les coordonnées de la zone de la fenêtre à rafraîchir ainsi que l'adresse de la zone mémoire (buffer) dans laquelle elle doit dessiner la fenêtre. *EndPaint* est appelée par l'application quand elle a fini de redessiner la fenêtre, pour signaler à la boîte à outils d'envoyer l'image de la fenêtre à l'écran. Les appels à *BeginPaint* et *EndPaint* vont toujours par paire. Notre librairie inclue deux stratégies pour obtenir l'image de la fenêtre à l'aide de ces fonctions.

La première stratégie intercepte l'appel à la fonction *EndPaint* et fait une copie du buffer mémoire dans lequel l'application a dessiné sa fenêtre. Cette copie est ensuite communiquée à *DiamondSpin*, qui lui applique les effets de rotation et des autres interactions disponibles, comme le pliage des fenêtres, avant de l'afficher sur la table interactive. Cette stratégie a deux inconvénients. Pour des raisons d'optimisation, *win32* ne conserve pas les zones du buffer mémoire correspondant à des zones de la fenêtre de l'application qui sont masquées (e.g. en dehors de l'écran, ou masquée par une autre fenêtre). En conséquence, l'image de la fenêtre peut ne pas être complète. Lors qu'une fenêtre est redessinée, notre librairie détecte donc si la fenêtre est partiellement masquée et utilise la deuxième stratégie si c'est le cas. Le deuxième inconvénient concerne le format du buffer mémoire. Il existe de nombreux formats pour représenter une image en mémoire, avec divers degrés de précision sur le nombre de couleurs, l'inclusion d'un canal pour la transparence (canal alpha), etc. La conversion entre certains formats est relativement lourde, et ne peut pas être effectuées à un rythme de rafraîchissement de 60 fois par seconde sur une fenêtre de grande taille. *DiamondSpin* utilise un format fixe, optimisé pour l'affiche en OpenGL. Or les applications et la boîte à outils *win32* utilisent divers formats lors de la création du buffer mémoire, dont certains sont lents à convertir dans le format de *DiamondSpin*. Cette première stratégie est celle illustrée dans la figure 11.

La deuxième stratégie intercepte l'appel à la fonction *BeginPaint* et change l'adresse de la zone mémoire dans laquelle l'application doit dessiner la fenêtre. L'avantage principal de cette stratégie est que notre librairie a un contrôle total sur ce buffer. Il est créé avec le même format que celui utilisé par *DiamondSpin*, ce qui permet une vitesse optimale dans le flux de données entre l'application et le système d'affichage de la table interactive. D'autre part, notre librairie ne tient pas compte des zones masquées de la fenêtre et conserve l'ensemble des données. En revanche, la fenêtre de l'application n'est plus rafraîchie dans le buffer fourni par *win32*. Cela peut être problématique pour certaines applications qui utilisent l'image de leur fenêtre pour

effectuer certains de leurs traitements (i.e. plutôt que de se baser sur leur représentation interne de leurs données). Toutefois, nous n'avons pas rencontré d'applications ayant ce problème lors de nos tests. Le buffer mémoire fourni par win32 est créé par win32 même s'il n'est plus utilisé, cette deuxième stratégie utilise donc plus de mémoire vive. Notre librairie intercepte également l'appel à la fonction *EndPaint*. Cet appel indique que l'application a fini d'utiliser le buffer, et qu'il peut maintenant être utilisé par DiamondSpin pour l'affichage sur la table interactive. Lors de *EndPaint*, notre librairie restore également l'adresse du buffer d'origine de win32, pour permettre à la fonction d'origine de la boîte à outils win32 de disposer d'une zone mémoire dans le format que la fonction d'origine *BeginPaint* lui a créé.

L'ensemble de cette librairie utilisée pour implémenter cette seconde preuve de concept modifie une cinquantaine de fonctions de la boîte à outils d'interface utilisateur win32. L'ensemble de ces fonctions est implémenté avec le langage de programmation C en 6000 lignes de code. Toutefois, une partie de ces fonctions ont été automatiquement générée par un script. En effet, leur seul but est de convertir des données entre les types de données C utilisés par win32 et les types de données Java utilisés par DiamondSpin, puis d'appeler la fonction équivalente dans l'autre boîte à outils, ce qui peut être automatisé. Nous avons simplement, pour ces fonctions, produit une liste d'équivalence entre les noms des fonction win32 et les noms des fonctions DiamondSpin.

1.4. Conclusion

Nous avons présenté six technologies qui peuvent permettre d'utiliser une application existante avec un environnement système différent, celui d'une table interactive. En effet, les tables interactives ne sont pas des ordinateurs de bureau, elles ont des fonctionnalités différentes, et donc les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau.

Nous avons évalué ces technologies de lien selon six axes : la structuration des données, la flexibilité, la performance, la compatibilité et la réutilisabilité, et la difficulté d'implémentation. La différence fondamentale entre ces technologies est la position du lien dans la chaîne de traitement entre l'application et le trio de dispositifs écran-clavier-souris pour lesquels elle a été conçue. La figure 12, en page suivante, résume nos cinq axes, nos six technologies et leurs évaluations. Nous avons également implémenté des prototypes ou des preuves de concept pour trois de ces technologies.

Les technologies ont divers avantages et inconvénients. Comme nous le présumons dans la présentation des métriques, le niveau de structuration des données et la flexibilité des modifications possibles sont liées. Toutes les technologies ont un niveau de flexibilité inférieur à leur niveau de structuration des données. Des données structurées sont requises pour permettre des modifications plus profondes de l'interface graphique de l'application existantes.

Les performances sont suffisantes pour une utilisation interactive pour toutes les technologies sauf la technologie utilisant des captures d'écran. La technologie utilisant une carte graphique virtuelle est une extension relativement simple de la technologie utilisant des captures d'écran, et elle résout le problème des performances du système. Si nous considérons les cinq autres technologies, en mettant de côté les captures d'écran, le critère de performance n'est pas décisif. Certes, des performances très élevées permettent de libérer plus de temps processeur pour les traitements de l'application elle-même. Toutefois, les applications que nous envisageons pour utiliser collaborativement sur une table numérique sont des applications interactives (qui utilisent la plupart de leur cycle d'exécution à attendre les entrées de l'utilisateur), et non des applications utilisant massivement la puissance de calcul de l'ordinateur pour, e.g., des traitements de données non interactifs.

La technologie « idéale » est probablement celle qui utilise tous les mécanismes que nous avons présentés. Ainsi, l'utilisation d'un langage de script pour interagir avec les applications existantes offrent la meilleure structuration des données et la meilleure flexibilité des modifications possibles. En revanche, ceci n'est compatible qu'avec certaines applications, et requiert que les caractéristiques de chaque application soient explicitement prises en compte par l'implémentation du lien logiciel. Un nombre limité d'applications qui peuvent bénéficier le plus de la flexibilité offerte

peuvent être utilisées avec cette technologie. Les autres applications peuvent être utilisées via la réécriture d'une boîte à outils d'interface homme machine, qui a une compatibilité plus élevée, et qui ne nécessite pas de programmer indépendamment le support pour chaque application envisagée. Enfin, les très rares applications qui n'utilisent pas de boîte à outils peuvent être liées à une carte graphique virtuelle, une technologie qui supporte 100% des applications existantes.

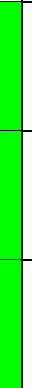
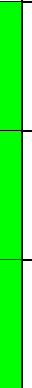
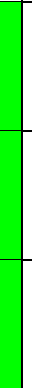
	Structuration des données	Flexibilité	Compatibilité et réutilisabilité	Performance	Difficulté d'implémentation
Capture d'écran (sortie graphique)	Bas 	Très bas 	Très élevé 	Bas 	Facile 
Carte graphique virtuelle (sortie graphique) *	Bas 	Bas 	Très élevé 	Élevé 	Moyen 
Clavier virtuel et souris virtuelle (entrée utilisateur)	Bas 	Très bas à bas 	Très élevé 	Élevé 	Facile 
Langage de script (entrée et sortie) *	Très élevé 	Élevé 	Bas 	Très élevé 	Moyen 
API d'accessibilité numérique (entrée et sortie)	Sortie : moyen  Entrée : élevé 	Moyen 	Moyen 	Moyen 	Moyen 
Réécriture d'une boîte à outils d'interface homme machine (entrée et sortie) *	Élevé 	Sortie : moyen  Entrée : élevé 	Élevé 	Très élevé 	Difficile 

Figure 12 : Technologies permettant d'utiliser une application existante avec un environnement différent du point de vue de son modèle de sortie graphique et d'entrée utilisateur. Un astérisque indique que nous avons implémenté un prototype ou preuve de concept pour cette technologie.

CHAPITRE 2

Interactions sur les tables interactives et logiciels existants

2.1. Introduction

Dans le précédent chapitre, nous avons présenté six technologies qui permettent de lier une application existante avec un environnement système différent, celui d'une table interactive. Les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau. Les technologies que nous avons présentées permettent faire le lien entre les techniques utilisées pour programmer des applications pour ordinateurs de bureau et les techniques utilisées pour programmer des applications pour tables interactives. Mais les applications pour tables interactives ne s'utilisent pas de la même façon que les applications pour ordinateurs de bureau non plus. Les interfaces utilisateur des applications existantes ont été conçues pour être utilisées par un seul utilisateur avec un ordinateur de bureau. Les technologies que nous avons présentées ne résolvent pas ce problème. Nous pouvons transporter les interfaces graphiques des applications existantes sur des tables interactives, mais elles peuvent souffrir de graves problèmes d'utilisabilité. Les systèmes de tables interactives se présentent sous un aspect matériel différent des ordinateurs de bureau ou des ordinateurs portables. Là où ces derniers utilisent un écran vertical pour la sortie graphique associé à un clavier et à une souris pour l'entrée, les tables interactives utilisent une surface horizontale qui fait à la fois office de périphérique de sortie graphique et de périphérique d'entrée tactile. Ces deux types de systèmes informatiques utilisent donc des types d'interactions différents pour permettre aux utilisateurs de manipuler le système. En revanche, les applications pour lesquelles ils sont utilisés présentent les mêmes fonctionnalités : éditer des documents, rechercher des informations,...

Collins et al. [21] ont exploré la façon dont les utilisateurs gèrent leurs fichiers sur une table interactive avec deux logiciels : un logiciel existant pour ordinateur de bureau, et un logiciel spécifiquement créé pour les tables interactives. Ils ont demandé à des paires de deux utilisateurs d'utiliser la table interactive pour accéder à des photographies d'attractions touristiques et d'hôtels en vue de planifier de manière collaborative un voyage. Dans la première condition, les utilisateurs se servaient du logiciel Windows Explorer (Explorateur Windows) de Microsoft, qui est intégré au système d'exploitation Microsoft Windows XP. Le logiciel était présenté exactement comme sur un ordinateur de bureau (figure 13, à gauche), avec les mêmes techniques d'interaction. En particulier, les fenêtres ne sont pas orientables, et l'ensemble du système ne gère qu'un seul point de contact simultané. Explorer a été configuré pour afficher chaque dossier dans une fenêtre séparée, et ouvrir les dossiers par simple touche (au lieu du double clic). Dans la seconde condition, les participants ont utilisé le logiciel Focus [20], qui a été développé spécifiquement pour les tables interactives (figure 13, à droite). Focus permet de parcourir des arborescences de fichiers de manière associative. Focus tire partie des caractéristiques des tables interactives, en permettant par exemple à plusieurs utilisateurs d'interagir en parallèle grâce au matériel multi-touche dont est dotée la table. Focus utilise également des techniques d'interactions spécifiquement conçues pour résoudre des problèmes d'interface

homme-machine inhérents aux tables interactives, comme le lancer inertiel de fichiers pour pallier à la grande taille de la table, ou la rotation de document pour permettre une interaction tout autour de la table. Les conclusions de cette étude sont pour le moins inattendues : les utilisateurs ont trouvé le logiciel existant (Explorer), non adapté aux tables interactives, plus facile à utiliser (à 63%), plus rapide pour effectuer la tâche (à 74%), et au final sont partagés sur leur préférence (47% préfèrent Explorer, et 53% préfèrent Focus). Les métriques mesurées par Collins et al. confirment ce résultat : les utilisateurs ont été en moyenne deux fois rapides pour compléter la tâche avec le logiciel existant qu'avec le logiciel spécifiquement créé pour la table interactive (8,8 minutes avec Explorer contre 16,7 avec Focus). En dépit de cela, les utilisateurs ont trouvé Windows Explorer peu adapté à la table interactive, se plaignant six fois plus d'encombrement de la table avec ce dernier.



Figure 13 : Les logiciels utilisés sur la table interactive : le logiciel existant Windows Explorer (à gauche), et le logiciel pour table interactive Focus (à droite). Le logiciel existant est plus rapide et plus facile à utiliser [21].

Cette étude nous amène à penser que notre contribution dans cette thèse, utiliser des logiciels existants adaptés à l'environnement et aux spécificités d'une table interactive, permettrait de combiner les avantages de chaque approche : la familiarité et la facilité d'usage des logiciels auxquels les utilisateurs sont habitués, tout en ajoutant de manière transparentes des techniques d'interaction conçues explicitement pour le travail sur une table interactive de manière plus efficace et plus intuitive. Les aspects physiques des tables interactives ne semblent pas poser de problème aux utilisateurs, mais l'absence de prise en compte du contexte du travail autour de la table gêne les utilisateurs : 53% d'entre eux préfèrent le logiciel développé pour les tables interactive même si celui-ci est plus lent et moins facile à utiliser que le logiciel qui n'est pas adapté aux tables interactives.

2.2. Scénarios

Nous l'avons vu avec l'exemple de Focus, il n'est pas facile de cerner les problèmes d'usage d'une table interactive sans se concentrer sur des tâches précises et des scénarios bien définis. Nous proposons plusieurs scénarios dans cette section pour illustrer certains des problèmes que pose l'utilisation directe de logiciels existants non adaptés aux tables interactives sur une table interactive. Nous présenterons ensuite des techniques d'interactions, envisageables pour les applications existantes, qui ont été créées en prenant en compte les spécificités des tables interactives. Le premier scénario concerne une tâche collaborative d'édition d'images avec le logiciel GIMP. Le second scénario une tâche collaborative d'édition d'une feuille de calcul avec le logiciel Microsoft Excel.

2.2.1. Édition d'images

Dans ce premier scénario, inspiré des très nombreux démonstrateurs de partages de photographies sur une table interactive, trois utilisateurs veulent réaliser une tâche d'édition d'image. Ils sont assis autour d'une table interactive sur laquelle ils utilisent l'application existante GIMP (The GNU Image Manipulation Program). Une capture d'écran de cette application telle qu'utilisée sur un ordinateur de bureau est présentée à la figure 14. Cette application utilise des palettes d'outils flottantes. Ces palettes d'outils sont indépendantes des documents ouverts. L'utilisateur peut les déplacer n'importe où sur sa surface de travail, aussi bien sur la surface occupée par le document que sur la surface libre autour du document, ou même sur la surface occupée par d'autres documents ou d'autres applications sans rapport avec la palette d'outils. Les palettes d'outils flottantes sont utilisées depuis longtemps, et donc bien maîtrisées par les utilisateurs. MacDraw offrait des menus détachables dès 1984. C'est ce principe qui a été repris et étendu par les interfaces transparentes telles que les Toolglass et les Magic Lenses [11], qui, dans le cadre des interfaces utilisateur pour ordinateur de bureau, se sont montrées plus efficaces que les interfaces non flottantes. Elles permettent notamment à l'utilisateur de rapprocher ses outils de prédilection de sa zone de travail actuelle, réduisant ainsi les trajets du dispositif de pointage. Ces derniers sont particulièrement consommateur de temps avec des dispositifs de pointages indirects, tels que la souris des ordinateurs de bureau.

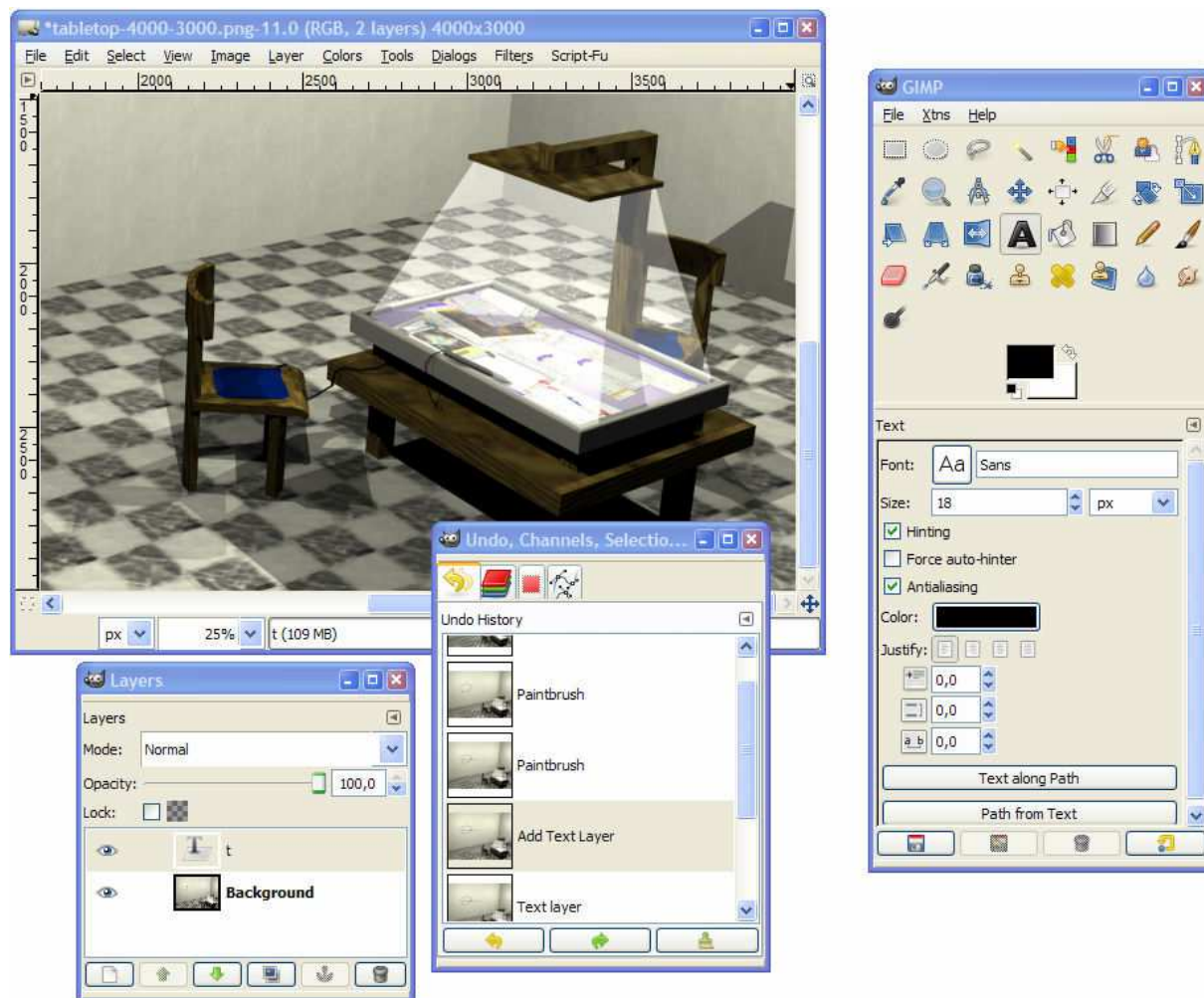


Figure 14 : l'application existante GIMP (www.gimp.org), utilisée sur un ordinateur de bureau, consiste en une fenêtre principale (en haut à gauche) par image et plusieurs palettes d'outils indépendantes. Les palettes d'outils peuvent être placées sur la fenêtre principale, à côté de la fenêtre principale, ou n'importe où sur l'écran de l'ordinateur.

Sur une table interactive multi-utilisateur, cette capacité des palettes d'outils flottantes à être repositionnées n'importe où sur la surface de travail peut se révéler gênante. Sur la figure 15, trois utilisateurs utilisent GIMP pour éditer des images. L'application est utilisée dans notre environnement DiamondSpin qui inclut un gestionnaire de fenêtres adapté aux tables interactives. Il permet notamment de tourner les fenêtres pour les orienter face aux utilisateurs situés tout autour de la table. Les palettes d'outils flottantes ne sont pas visuellement associées au document qu'elles affectent. Dans le cadre d'un utilisateur unique sur un ordinateur de bureau, celui-ci ne travaille que sur un seul document à un moment donné. Les palettes d'outils affichées à ce moment sont les palettes d'outils concernant ce document. L'application existante n'a donc pas prévu de mécanisme pour indiquer la relation entre la palette d'outils et son document. Dans le cadre de plusieurs utilisateurs travaillant autour d'une table interactive sur plusieurs documents, plusieurs jeux de

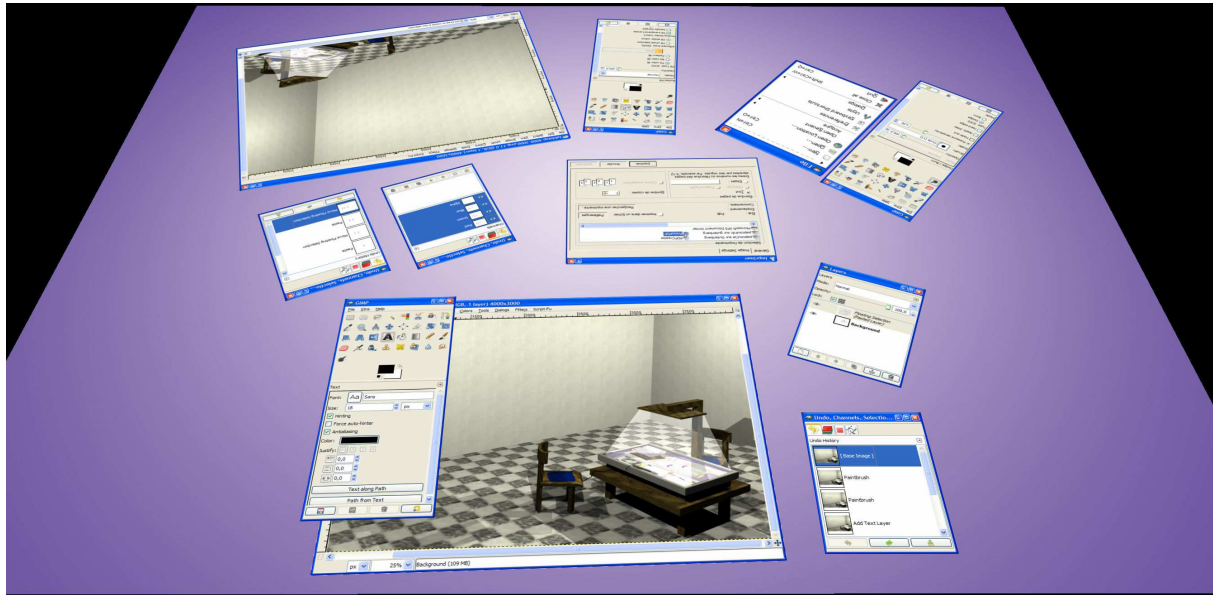


Figure 15 : l'application existante GIMP (www.gimp.org) utilisée par trois utilisateurs sur une table interactive. Deux utilisateurs (situés sur la figure en bas et en haut à gauche) travaillent chacun sur une image dans une fenêtre principale et des palettes d'outils. Le troisième utilisateur (en haut à droite) n'a que des palettes d'outils, peut-être configure-t-il l'application ou recherche-t-il une image à éditer. Aucune des palettes d'outils flottantes ni des boîtes de dialogue ne sont clairement associées au document qu'elles affectent ou à l'utilisateur qui les possède.

palettes d'outils sont affichés sur la table. L'utilisateur ne peut pas savoir quelle palette d'outils affecte quel document.

2.2.2. Édition d'une feuille de calcul

Dans ce second scénario, nous nous inspirons de l'une des applications les plus utilisées en bureautique, le tableur numérique. Ce type d'applications est omniprésent dans le monde du travail. Les tableurs numériques, tels que Microsoft Excel, sont utilisés aussi bien pour leur valeur intrinsèque que comme plateforme pour des applications tierce partie de traitement de données. Souvent, plusieurs utilisateurs doivent intervenir sur une même feuille de calcul du tableur. Ainsi, nous avons noté plusieurs scénarios d'usage, dans lesquels les différents utilisateurs se réunissent autour d'un ordinateur commun ou se partagent un ou plusieurs fichiers Excel de façon asynchrone, par exemple par e-mail. Les tâches que nous avons relevées incluent créer collaborativement un planning, rédiger une liste en commun, gérer et produire le résultat de jurys ou de réunions, ou remplir des formulaires ou des rapports impliquant plusieurs personnes.

Nous détaillons un de ces scénarios. Quatre utilisateurs se réunissent autour d'une table pour éditer collaborativement un formulaire représentant l'état de divers éléments sur plusieurs années. Ils utilisent le tableur Microsoft Excel pour saisir leurs données dans une feuille de calcul sur plusieurs lignes et plusieurs colonnes. Excel

	A	B	C	D	E	F	G	H
1		Etats						
2						Prévision		
3		Unité	2008	2009	2010	2011		
4	Articles	pièces	2	3	1	2		
5	Ateliers	pièces	4	2	0	1		
6	Thèse	pages	0	30	70	0		
7	Programmes	lignes de code	5000	5500	3800	4200		
8	Voyag			1				
9								
10								
11								
12								
13								
14								

Figure 16: Le formulaire du scénario 2, ouvert dans le logiciel Microsoft Excel, dans la configuration classique d'un ordinateur de bureau. Le document est rempli avec les mêmes données que dans la figure 17.

utilise un système d'interface à documents multiples (MDI), dans lequel une unique fenêtre est utilisée pour afficher l'ensemble des documents [68], présenté sur la figure 16.

Excel permet de créer jusqu'à quatre vues d'un document (figure 16, la séparation entre les quatre vues est surlignée en rouge), mais ces vues ne sont pas adaptées pour être utilisées par plusieurs utilisateurs. Les vues ne sont pas totalement indépendantes les unes des autres. L'ensemble des quatre vues n'a que quatre degrés de liberté, représentées par quatre barres de défilement. Pour être indépendantes les unes des autres, elles devraient avoir huit degrés de liberté. Ainsi, si chaque vue est assignée à un utilisateur (comme sur la figure 17), l'utilisateur rouge (en haut à gauche) *doit* travailler sur les mêmes lignes que l'utilisateur placé à sa gauche (utilisateur bleu, en haut à droite) et les mêmes colonnes que l'utilisateur placé à sa droite (utilisateur violet, en bas à gauche). De même, l'utilisateur bleu, qui doit donc travailler sur les mêmes lignes que l'utilisateur rouge à sa gauche, doit également travailler sur les mêmes colonnes que l'utilisateur à sa droite (utilisateur vert, en bas à droite). Chacun des quatre utilisateurs est influent sur la partie de la feuille de calcul que peuvent voir les utilisateurs placés à côté de lui.

répliqués pour chaque vue, plusieurs utilisateurs ne peuvent pas les utiliser simultanément.

2.3. Conclusion

Ces limitations ne sont pas surprenantes. Microsoft Excel n'a pas été conçu pour être utilisé par plusieurs utilisateurs sur une table interactive. De même, l'usage, par l'application GIMP, de plusieurs fenêtres pose des problèmes d'utilisabilité sur une table interactive. Même si les technologies que nous avons présentées au chapitre précédant permettent de lier des applications existantes avec l'environnement système des tables interactives, les applications en question ne sont pas automatiquement adaptées aux usages sur des tables. Pourtant, l'étude du cas de Windows Explorer vs. Focus l'a montré, la moitié des utilisateurs sont prêts à passer outre ces problèmes d'utilisabilité. D'autre part, le logiciel existant reste plus facile à utiliser et plus rapide pour accomplir la tâche. Nous pensons que les logiciels existants, couplés à des techniques d'interactions conçues en prenant en compte les spécificités des tables interactives, peuvent combiner les avantages des deux approches. Nous nous sommes intéressés à des techniques d'interactions qui peuvent être intégrées de manière automatique et transparente pour l'application, à des applications existantes. Un exemple d'intégration automatique et transparente serait de remplacer les menus déroulants par des menus circulaires. Les fonctionnalités sont identiques et interchangeableables. Une des principales différences entre les tables interactives et les ordinateurs de bureau est dans la gestion de l'espace. L'espace d'une table interactive est partagé par plusieurs utilisateurs, celui d'un ordinateur de bureau ne l'est pas. L'espace d'une table interactive est horizontal et peut être vu selon plusieurs orientations autour de la table, celui d'un ordinateur de bureau est vertical et n'est utilisé que dans une orientation. L'espace d'une table interactive peut avoir des zones qui sont hors de portée d'un utilisateur, celui d'un ordinateur de bureau est entièrement accessible avec la souris. Dans le chapitre suivant, nous proposons plusieurs techniques d'interaction pour la gestion de l'espace sur les tables interactives.

CHAPITRE 3

Gestion de l'espace sur les tables interactives _____

3.1. Introduction

Plusieurs travaux rapportent que l'espace disponible sur une table interactive est de trop petite taille, et limitant pour les utilisateurs. Nous nous proposons d'observer la gestion que les utilisateurs font d'un espace contraint lors de leur utilisation d'une table traditionnelle avec les documents physiques. Scott *et al.* [103] rapportent que les utilisateurs utilisent l'espace disponible de manière opportuniste. De plus, ils n'hésitent pas à annexer de l'espace en dehors de la surface de la table si celle-ci n'offre plus d'espace libre, tel que des chaises, des boîtes et autres conteneurs, d'autres tables proches voire même le sol. Scott *et al.* notent que plus l'espace de stockage est loin de la zone principale de travail, moins les éléments qui y sont placés sont utilisés. Les utilisateurs se servent de ces zones comme lieu de stockage à moyen et long terme. Les éléments similaires sont stockés par les utilisateurs sous forme de piles d'éléments organisées de manière informelle au fur et à mesure que les éléments sont découverts sur la surface de travail. Certaines piles sont créées sur des surfaces non fixes, telles que des boîtes, ce qui permet aux utilisateurs de déplacer la pile dans son ensemble. Scott *et al.* ont notamment observé des utilisateurs qui déplaçaient les piles de la périphérie de l'aire de travail jusqu'au centre de la table pour récupérer un élément ou comparer les éléments de la piles avec les éléments de la surface de travail.

Malone [71] décrit la pile comme une unité structurelle dans l'organisation de documents sur un bureau. Il oppose la pile au fichier (*file*), dans le sens d'un contexte de documents physiques. Toutefois, la caractérisation d'un fichier qu'il utilise [119], à savoir un groupe d'instances de documents du même type pour lequel un ou plusieurs ordres (alphabétique, chronologique,...) peut être défini, correspond à ce que nous appelons plus communément un dossier dans ce domaine³. Dans la suite de cette discussion, nous utiliserons donc le terme dossier. Dans un bureau, les dossiers contiennent les documents dans des containers spécialisés (chemises, bacs de rangements,...) qui prévoient une zone pour les labelliser. Certains containers ont une fonction précise. Malone note l'usage de bacs « *in* » et « *out* » pour contrôler le flux de documents passant dans un bureau. Une analogie intéressante peut être faite avec l'usage, de nos jours, de l'email et des dossiers « boîte de réception ».

Au contraire, les piles regroupent des documents mais l'ensemble n'a pas de titre. Les éléments contenus dans une pile peuvent également ne pas avoir de noms. Malone donne l'exemple de notes, de divers papiers annotés, de listes de tâches informelles, etc. Dans le cas d'un système informatique, nous pouvons faire une analogie avec le cas d'un document non sauvegardé, d'un fragment de donnée non représenté par un fichier, comme un e-mail, ou de ressources externes, comme une adresse de site Internet, etc. Les éléments d'une pile ne sont pas ordonnés, et il n'existe pas d'ordre particulier que l'on puisse dériver des documents. Une pile de photographies, par

³ Au contraire, Malone définit les *folders* comme les éléments individuels contenus dans les *files*.

exemple, n'est pas classée, triée ou ordonnée comme un album de photographies pourrait l'être par date, par lieu ou par thème. Malone note que le mécanisme usuel de création d'une pile tend à insuffler un arrangement chronologique dynamique à la pile : les documents les plus récents tendent à être ajoutés en haut de la pile, tandis que les documents plus anciens et moins utilisés progressent vers le bas de la pile. Cet ordre informel ne peut pas être automatisé par le système informatique, contrairement au tri par nom, date, taille, et les autres attributs qui sont associés aux fichiers numériques. En revanche, il est particulièrement intéressant de noter comment cet ordre fait appel à la mémoire spatiale des utilisateurs quand il s'agit de retrouver un document dans une pile.

De l'observation de la façon dont les utilisateurs organisent leurs documents physiques dans les lieux de travail habituels, comme leur bureau, nous pouvons tirer des leçons sur la façon dont les utilisateurs interagissent avec les documents et l'espace pour informer la création d'interactions pour les tables interactives. Hartmann *et al.* [43] remarquent ainsi qu'une stratégie possible pour gérer le manque de place sur un bureau est de ranger une partie des documents, et autres artefacts de la surface de travail, dans un tiroir. Les tiroirs de bureau sont organisés de façon spatiale pour permettre de compartimenter les différents éléments à stocker dans des zones offrant des propriétés de « cherchabilité », et de contrôle d'accès. Le contrôle d'accès, Hartmann *et al.* notent, est fourni par l'accès physique à la poignée du tiroir, ainsi que par le degré d'ouverture du celui-ci (légèrement ouvert pour ne révéler qu'une partie du contenu, ou complètement ouvert pour un accès total aux documents qu'il contient). Ils proposent ainsi un mécanisme physique de tiroirs, qui se comporte comme un périphérique d'entrée de l'ordinateur table interactive, et contrôle un logiciel qui permet le stockage de documents numériques dans une zone hors surface de la table (figure 18). Quand le tiroir physique est ouvert, les documents numériques qu'il « contient » sont affichés sur la table interactive.



Figure 18 : Des « tiroirs de bureau » sont placés sous cette table interactive DiamondTouch. Les tiroirs, physiques, « contiennent » des documents numériques, un exemple d'interaction tangible [43].

Hartmann *et al.* observent des usages intéressants de leur système, par exemple l'utilisation d'un tiroir supplémentaire toujours ouvert comme d'une poubelle où les documents qui n'intéressent pas les utilisateurs sont placés d'un simple geste. Ils associent cela à l'utilisation, sur un bureau traditionnel, d'un « *junk drawer* » (voir figure 19), un tiroir où l'on range une collection d'objets qui n'ont pas de place dans le

système de catégorisation que l'on utilise, pêle-mêle avec les objets que l'on devrait jeter mais dont l'on ne peut se résoudre à se séparer définitivement [115]. Cet usage émergent où l'utilisateur étend l'analogie qui a présidé à la conception de la technique d'interaction nous intéresse particulièrement. Ainsi, quand Malone [71] postule que les piles pourraient permettre de rendre les ordinateurs plus naturels, il nous semble, à nous, observateur de l'interaction entre l'homme et la machine, naturel de chercher à la rendre la plus métaphorique possible.



Figure 19 : Les utilisateurs utilisent les métaphores pour reproduire leurs usages existants avec les objets inspirant la métaphore, comme ici un *junk drawer* [115]

3.2. Métaphore du papier

Les métaphores sont utilisées, en interaction homme-machine, pour aider les utilisateurs à appréhender le fonctionnement d'un nouveau système plus rapidement en y appliquant des connaissances d'un système qu'ils connaissent déjà [4]. Dans le domaine du travail collaboratif autour d'une table, le papier joue un rôle particulièrement important. Des documents en papier peuvent être transportés facilement. Ils peuvent être rangés, classés, reliés, découpés, stockés. Le papier lui-même est un support modifiable qui peut être annoté, effacé, ou au contraire contenir de l'information sur des durées importantes. Le papier est utilisé depuis des millénaires comme support de la connaissance humaine. Des manuscrits de l'Antiquité livrent encore leurs contenus aujourd'hui. Les documents en papiers peuvent intégrer facilement du texte avec des images, graphiques, dessins, et combiner des parties imprimées et manuscrites. La métaphore du papier existe depuis longtemps avec divers degrés de sophistication. En 1981, Xerox faisait la publicité de sa station de travail Xerox Star en mettant en avant la souplesse de son système d'édition de document et en comparant ce système à du papier (figure 20).

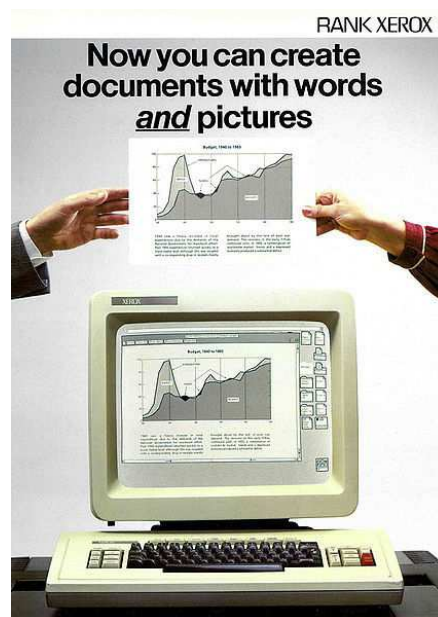


Figure 20 : Publicité pour le Xerox Star (1981). Le contenu d'un document numérique affiché à l'écran est comparé avec un document imprimé sur du papier.

En 2001, le projet *LivePaper* [95] propose d'augmenter des feuilles de papier physiques avec des informations numériques (figure 21). Les feuilles de papiers, suivies par un système de reconnaissance optique, sont localisées dans l'espace 3D. Un projecteur, fixé au plafond, est utilisé pour afficher les informations numériques sur les feuilles de papier. La zone dans laquelle ce système est actif est limitée, le projecteur et la caméra ne couvrent que la surface d'un bureau.

Ce système utilise des documents (papier) existants. Divers documents sont augmentés avec diverses informations. Des plans d'architectures sont augmentés avec une vue en trois dimensions du bâtiment qui « sort » de la feuille de papier. Deux systèmes *LivePaper* distants peuvent être utilisés pour annoter coopérativement, à distance, deux exemplaires d'un document. Certains documents déclenchent des applications comme un lecteur multimédia, et affiche une interface qui permet à l'utilisateur d'interagir avec les données numériques depuis le document papier.

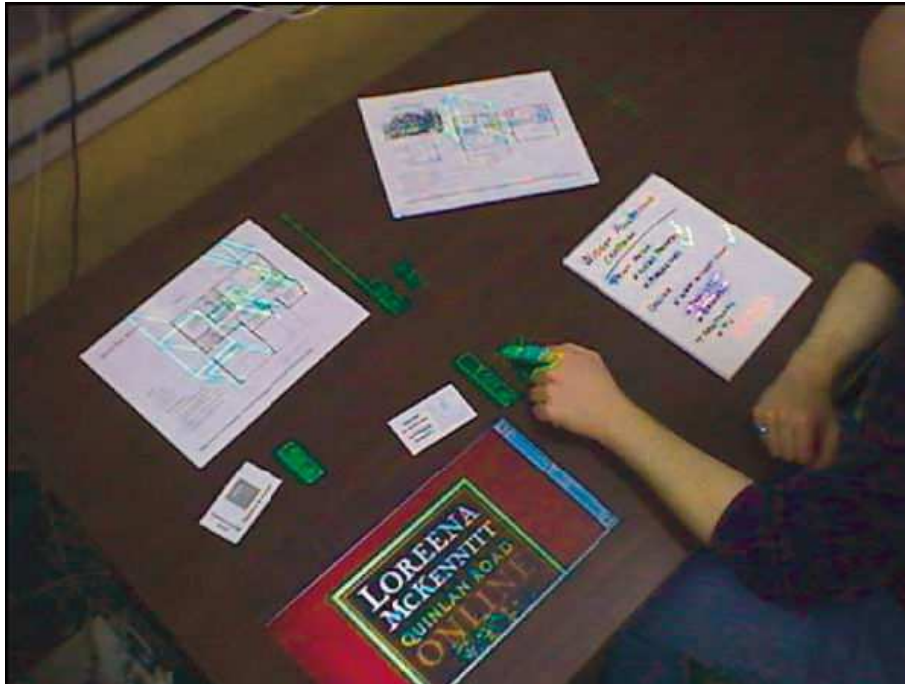


Figure 21 : Le système *LivePaper* augmente des feuilles de papiers ordinaires avec des informations numériques projetées [95].

De manière similaire, le projet *PaperWindows* [46] est un système de réalité augmentée qui propose de faire sortir les fenêtres des ordinateurs pour les transférer sur des feuilles de papier physique (figure 22). Ce système utilise une caméra et un vidéo projecteur de façon similaire à *LivePaper*. En revanche, les feuilles de papier sont originellement vierges. L'ensemble du contenu est numérique. Il inclut tous les éléments de l'interface des applications pour ordinateur de bureau. L'utilisateur peut interagir avec ces éléments en utilisant son doigt pour contrôler le pointeur. Un ensemble de gestes correspondant à l'usage de feuille de papier (tourner la page, empiler des pages,...) est également utilisé pour déclencher des actions spécifiques comme se déplacer dans l'historique de l'application du navigateur web.

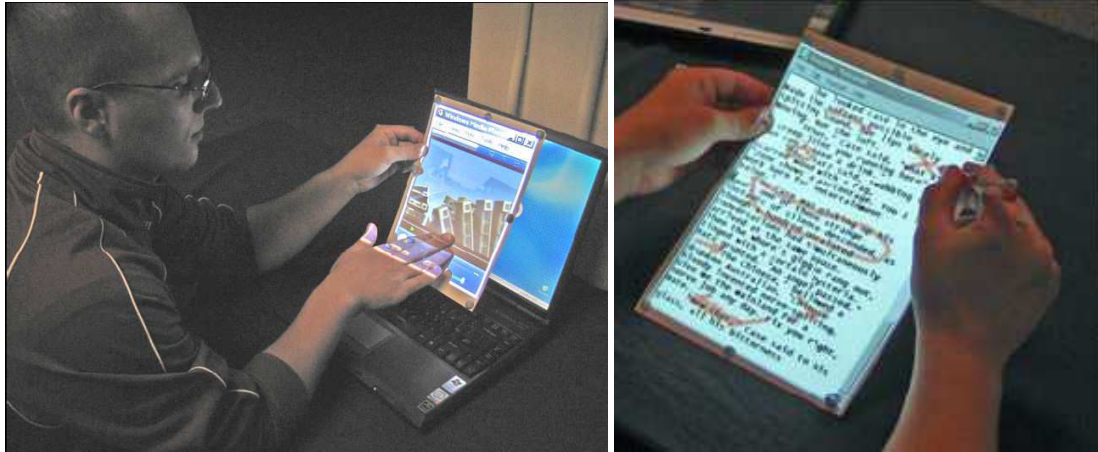


Figure 22 : Le système *PaperWindows* transfère les fenêtres d'applications sur des feuilles de papiers physiques. L'utilisateur peut continuer d'interagir avec les fenêtres en utilisant son doigt et les annoter avec un stylo [46].

Dans [59], Alan Kay revient longuement sur son expérience au Xerox PARC sur la conception des premières métaphores visuelles de l'interaction homme-machine. Il met en avant que le mot « métaphore » n'est pas une bonne métaphore du but recherché. Ainsi dans la métaphore du papier, il pense que le papier métaphorique ne doit pas être aussi difficile à effacer que du papier réel. Il propose le terme « illusion utilisateur » (*user illusion*) à la place : un système se rapprochant du théâtre. L'utilisateur sait que l'objet métaphorique n'est pas l'objet réel. Il s'attend à des comportements additionnels ou différents, que Alan Kay qualifie de comportements *magiques*. Ainsi, dans la métaphore, ou l'illusion, du papier, l'utilisateur s'attend à une gomme magique, qui ne laisse pas de trace et peut effacer sélectivement une partie seulement du document. L'important est de rendre cette magie compréhensible et prévisible.

La métaphore du papier est à mettre en contraste avec les opinions que les utilisateurs ne pourront jamais, voire ne veulent pas, ne plus utiliser l'impression papier de leurs documents de plus en plus numériques. Sellen *et al.* [106], dans *The Myth of the Paperless Office* (Le Mythe du bureau sans papier), arguent que le papier ne sera pas supplanté par les systèmes informatiques. Ils listent les avantages du papier : flexible, léger, lisible. Un livre, par exemple, est plus que la somme de ses pages. La couverture et la reliure donne des informations sur la qualité du livre. L'épaisseur informe l'utilisateur, sans que celui-ci n'en soit forcément conscient, sur la taille de l'ouvrage. En réponse à ces arguments, Hong *et al.* [47] proposent un système de livre virtuel en fausse 3D qui indique la progression dans le livre de façon visuelle (figure 23), grâce à l'épaisseur de la tranche du livre. Ils ajoutent ensuite des interactions « magiques », comme des tranches interactives : quand l'utilisateur déplace le curseur de la souris sur la tranche, le numéro de page correspondant à cet emplacement s'affiche.

ordinateurs de bureau se sont heurtées à l'inefficacité de la souris pour interagir en parallèle avec plusieurs strips et au manque de continuité entre le système papier et le système numérique, rendant la gestion du trafic aérien plus complexe [69].

Le niveau de détails visuels de la représentation du papier virtuel varie grandement. Denoue *et al.* [23] mettent en œuvre un algorithme sophistiqué pour rendre le papier le plus réaliste possible. Les utilisateurs interagissent avec le papier virtuel sur un écran tactile vertical, représentant un panneau d'affichage de type *bulletin board* en liège (figure 25). Les utilisateurs peuvent plier les feuilles de papier virtuel sans contrainte. Ils peuvent par exemple froisser complètement une feuille de papier. Des punaises virtuelles, en rouge sur la figure, peuvent également être manipulées.



Figure 25 : Du papier virtuel au rendu réaliste. Un système de maillage de points de référence et d'équation de force permet de plier le papier sans contrainte [23].

A l'opposé, le livre virtuel [47] mentionné précédemment (voir la figure 23 à la page précédente), n'a pas une surface courbe, avec un rendu en trois dimensions. Les auteurs mentionnent spécifiquement qu'un tel rendu serait plus réaliste, mais aussi rendrait plus difficile la lecture du texte à l'écran. A la place, le livre est représenté sous la forme d'une pseudo 3D constituée uniquement de parallélogrammes. Une animation est utilisée pour tourner les pages, mais elle ne permet pas d'interagir avec le papier de façon libre.

3.3. Fentes

Nos premiers travaux sur la métaphore du papier portent sur l'utilisation d'accessoires virtuels pour interagir avec le papier virtuel. Nous observons que des feuilles de papiers sont souvent insérées dans des fentes (figure 26). Nous postulons que des fentes virtuelles, sur une table interactive, inciteraient les utilisateurs à y glisser des documents. Les effets de telles fentes (ou *slots* en anglais) sont variés.



Figure 26 : Les feuilles de papiers sont souvent insérées dans des fentes. De gauche à droite : une boîte aux lettres, un télécopieur, une déchiqueteuse et un ordinateur à carte perforée (IBM Type 31).

La transmission de documents entre les utilisateurs d'une table est une interaction qui peut bénéficier grandement de cette métaphore. Les tables interactives étant de plus en plus grandes, transmettre un document à un autre utilisateur devient problématique. Les interactions comme le jeter de documents, avec un modèle physique de l'inertie et des frottements, nous semblent approximatives et peu pratiques. Lors de nos essais d'inertie, les utilisateurs ont eu des difficultés à placer le document à l'endroit désiré. Les problèmes viennent autant de la déviation angulaire (une légère déviation lors du lancer produit un large écart à grande distance), que du dosage de la force nécessaire pour parcourir la distance voulue et ni plus, ni moins.

En revanche, les utilisateurs transmettent quasi quotidiennement des documents papier en les glissant dans une boîte aux lettres. Sur une table interactive, nous proposons une fente de boîte aux lettres virtuelle pour déplacer un document à un autre endroit, inaccessible, de la table interactive (figure 27). Par extension, la fente du télécopieur ou du fax permet d'envoyer une copie du document à un autre endroit tout en conservant le document à son emplacement actuel. Cette interaction peut être étendue à la communication entre plusieurs tables interactives reliées par un réseau informatique, de manière similaire aux transfolders [110].

La position des fentes, directement « découpées » sur la surface de travail de la table horizontale, diffère de celles des fentes physiques habituelles, qui sont découpées sur des supports verticaux, tels que des murs. Il y a notamment un risque que les utilisateurs activent de manière involontaires une fente en déplacement un document par-dessus celle-ci. Nous avons ajouté un bouton (en vert sur la figure 27) pour activer la fente. Quand le bouton n'est pas pressé, la fente est inactive et la surface qu'elle occupe peut être utilisée comme le reste de la table. La fente de réception peut

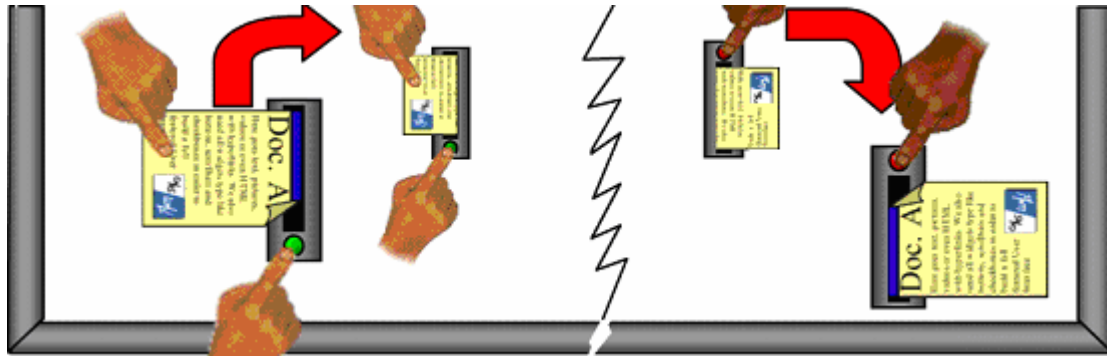


Figure 27 : Une fente virtuelle permet de transmettre des documents sur une table interactive.

également être activée à la demande (bouton rouge sur la figure 27) pour ne pas perturber le travail de l'utilisateur qui reçoit le document.

Pendant la transmission, le document peut être modifié par le système. Dans l'exemple de la figure 27, le document est retourné pendant la transmission pour pouvoir être lu directement par l'utilisateur qui le reçoit. L'interaction devient magique, au sens décrit plus haut, en cela que la fin du document apparaît sortir du fax virtuel de réception avant qu'elle ne soit entrée dans le fax d'émission.

Pour adresser le document à la bonne personne, ce qui est un problème important avec les interactions de type jeter inertiel, les fentes peuvent soit être associées deux à deux, soit augmentées d'une interaction pour désigner la fente de sortie. L'association deux à deux des fentes peut se faire par exemple à l'aide d'un code couleur.

Nous avons utilisé des fentes dans le cadre d'une expérience utilisateur sur la créativité. Dans cette expérience, quatre participants cherchent à trouver le plus d'idées possible sur un thème donné. Nous avons comparé trois conditions : avec des post-it en papier et des stylos physiques, avec des post-it virtuels sur une table interactive, et avec des post-it virtuels augmentés d'interactions spécifiques aux tables interactives tel que les fentes ou le pliage (le pliage est décrit ci-après). Les fentes permettent de stocker les post-it créés par les participants. En effet, lorsque des centaines de post-it s'accumulent sur la table, la gestion de l'espace devient difficile pour les utilisateurs. Par ailleurs, la réserve de certains utilisateurs peut les mener à l'autocensure. Nous avons introduit les fentes virtuelles dans la troisième condition pour répondre à ces deux besoins : une fente est disponible devant chaque utilisateur pour y glisser des post-it, quand l'utilisateur veut faire de la place ou quand il veut cacher un post-it qu'il a créé. Les post-it sont envoyés dans un espace de stockage collectif et réapparaissent tous ensemble lors de l'étape suivante de la séance de créativité (l'étape de catégorisation). Afin d'éviter le stockage non intentionnel, nous avons paramétré empiriquement la cinématique de sorte que les gestes trop rapides fassent passer les post-it au-dessus de la fente et non dedans.

L'expérience a résulté en de meilleures performances pour la condition papier et stylo. De précédents travaux ont montré que la production d'idées pouvait être

influencée par la stimulation cognitive [27, 28, 85] ainsi que par la comparaison sociale [5, 42, 75, 88]. La condition papier et stylo était supposée générer une faible stimulation cognitive (les participants ne voyaient pas le contenu des post-it des autres participants pendant la production d'idées) et une forte comparaison sociale (car ils voyaient physiquement le nombre de post-it remplis et empilés par les autres). A l'inverse la condition table interactive avec les interactions de fentes a dû provoquer plus de stimulation cognitive (les participants voyaient les post-it des autres pendant l'édition, avec un texte et des dessins plus faciles à traiter que de l'écriture manuscrite quand ils sont vus à l'envers) mais une comparaison sociale inférieure (une fois les post-it créés ils pouvaient être insérés dans les fentes et il n'y avait alors plus de feedback sur le nombre de post-it créés par chaque participant). Nous supposons donc que le facteur principal a été la comparaison sociale. L'autre hypothèse pour expliquer nos résultats provient de nos observations des sessions : avec la table interactive enrichie, les utilisateurs ont passé plus de temps à éditer des post-it qu'en condition papier et stylo. La saisie de texte était plus lente qu'avec un crayon, et certains utilisateurs ont perdu du temps à « jouer » avec certaines de nos nouvelles interactions [12].

Un autre aspect de la métaphore des fentes qui nous semble intéressant est leur affordance à opérer des traitements sur les documents. Parmi nos exemples de la figure 26, une déchiqueteuse à papier détruit les documents qui sont insérés dans sa fente. L'ordinateur à cartes perforées utilise les informations présentes sur le document inséré dans sa fente pour définir les traitements à appliquer. Il s'agit d'une certaine façon d'une fente extrêmement générique qui se comporte de diverses façons selon la nature du document inséré. D'autres appareils utilisent des fentes, par exemple un lecteur de disques compacts (CD) produit de la musique à partir des informations présentes sur le disque qui est inséré dans le lecteur⁴. Une fente pourrait être utilisée pour effacer des annotations d'un document, ou au contraire pour révéler les annotations d'autres utilisateurs de façon similaire à une Magic Lens [11].

⁴ Certains modèles de lecteurs utilisent un autre dispositif au lieu d'une fente. Par exemple, la plupart des chaînes Hi-Fi utilisent un plateau à CD, tandis qu'avec la plupart des lecteurs de CD portables, le disque est directement posé dans l'appareil. La plupart des autoradios, en revanche, utilisent une fente.

3.4. Pliage

Un des aspects qui distingue le papier des autres supports d'informations est sa capacité à être plié. Après la lecture, le pliage du papier est sans doute une des interactions les plus communes que les utilisateurs font des feuilles de papier. Plusieurs exemples de papiers pliés sont représentés dans la figure 28. Sur la première ligne, le document en papier est plié dans un but utilitaire opportuniste (de gauche à droite) : marquer une page d'un livre, comparer deux graphiques présents sur deux pages d'un livre, et réduire la taille d'un document pour le mettre dans sa poche en le pliant en quatre, ou dans un étui à posters en le roulant sur lui-même. Sur la seconde ligne, le pliage fait partie de l'interaction prévue avec le document en papier (de gauche à droite) : un carton d'invitation livré plié et cacheté, un origami, l'art japonais du pliage de papier (de « ori » (plier) et « kami » (papier)), et un livre pop-up ou livre animé contenant des éléments en papier plié qui se déploient à l'ouverture d'une double page.



Figure 28 : Le pliage du papier fait partie intégrante des interactions que l'utilisateur peut avoir avec celui-ci.

Cet usage du papier lui donne une dimension supplémentaire : le document n'est plus un plan en deux dimensions, il occupe maintenant un espace en trois dimensions. Nous observons actuellement une évolution de deux à trois dimensions dans les environnements graphiques des ordinateurs de bureau. Certains de ces nouveaux environnements graphiques prenant en compte la troisième dimension se base sur des métaphores du monde physique. Ainsi, le pliage des fenêtres informatiques a été proposé en 2001 [6] pour gérer les situations de superposition de fenêtres qui se

produisent avec la métaphore du bureau à deux dimensions [94]. Plusieurs techniques d'interaction sont basées sur le pliage des fenêtres. La technique *Fold'n'drop* [25] permet aux utilisateurs de retourner les fenêtres de son choix avec un geste de va et vient du pointeur de la souris. Ce geste est combiné avec l'interaction traditionnelle du glisser-déposer (drag'n'drop) pour permettre de déposer une icône sous plusieurs fenêtres superposées. En effet, dans l'interaction de glisser-déposer, la source et la destination du glisser-déposer doivent être visible à l'écran en même temps. Cette interaction ne peut pas être utilisée entre deux fenêtres superposées. *Fold'n'drop*, en permettant de révéler sélectivement des fenêtres masquées (y compris des fenêtres masquées par la fenêtre source), augmente le nombre de destinations possible pour le glisser-déposer (figure 29, gauche). Lorsque le glisser-déposer est terminé, les fenêtres se déplient automatiquement et retrouvent exactement leur emplacement d'origine.

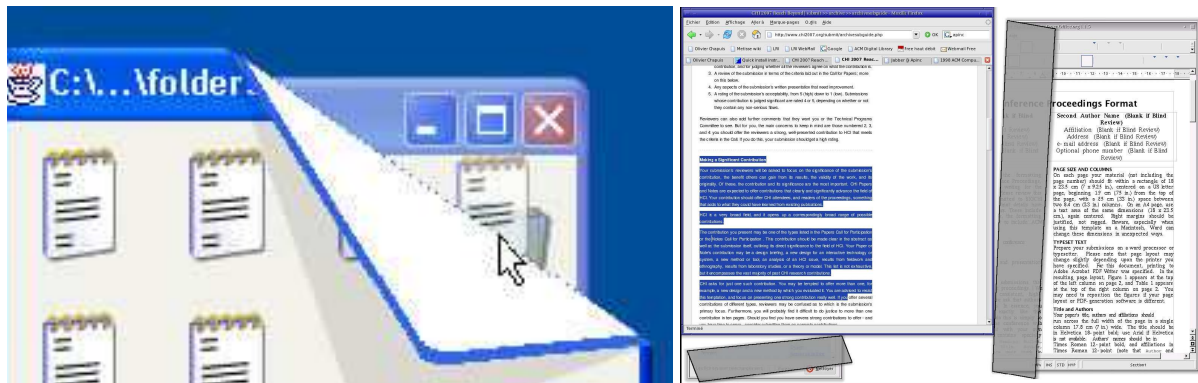


Figure 29 : Le pliage des fenêtres, dans le cadre d'un ordinateur de bureau, permet d'accéder à des fenêtres masquées par la superposition d'autres fenêtres. A gauche, *Fold'n'drop* [25], à droite des fenêtres repliées sur elles-mêmes pendant un copier-coller [17].

Le pliage des fenêtres a également été proposé pour permettre le copier-coller entre des fenêtres superposées. Les environnements d'ordinateurs de bureau utilisent la notion de fenêtre active, la fenêtre au premier plan avec laquelle l'utilisateur interagit. Il s'agit du centre d'intérêt principal de l'utilisateur. Dès que l'utilisateur commence à interagir avec une autre fenêtre, celle-ci devient la nouvelle fenêtre active, même si l'utilisateur veut continuer de travailler sur le même centre d'intérêt principal. L'environnement des ordinateurs de bureau traditionnels ne permet pas de gérer un centre d'intérêt secondaire, temporaire de l'utilisateur. Chapuis *et al.* [17] donnent l'exemple de l'édition d'un document (centre d'intérêt principal) qui requiert de copier-coller un passage depuis un navigateur web (centre d'intérêt secondaire et temporaire) dans l'éditeur de document. Le changement de fenêtre active oblige l'utilisateur à effectué des tâches de gestion de fenêtres qui vont perturber le flux de son travail. Chapuis *et al.* proposent d'introduire la notion de fenêtre temporairement active, pendant la durée d'une interaction de type sélection à la souris (par glisser-déposer). Sur la figure 29 (droite) l'éditeur de texte est la fenêtre principale, mais pendant que l'utilisateur sélectionne du texte dans le navigateur web (la fenêtre secondaire), elle est repliée sur elle-même. La fenêtre en bas de l'écran n'est pas une

fenêtre active et est repliée sur elle-même de façon similaire pour permettre au centre d'intérêt secondaire de l'utilisateur d'être entièrement visible. Quand l'utilisateur a fini d'interagir avec la fenêtre secondaire, les fenêtres se déplient automatiquement et l'utilisateur retrouve son centre d'intérêt principal.

Le point commun à ces deux interactions est l'aspect temporaire du pliage des fenêtres. Les fenêtres se plient pour permettre une interaction spécifique : le glisser-déposer, la sélection de texte et le copier-coller. Or nous observons dans les exemples de la figure 28, que le dépliage du papier physique n'est pas automatique, et encore moins conditionné à la fin d'une autre interaction. Nous proposons d'explorer le pliage permanent ou, du moins, durable des documents dans le cadre de l'interaction avec une table interactive.

Lorsqu'ils ont été présentés le système de *Fold'n'drop*, plusieurs utilisateurs ont remarqué en plaisantant qu'ils « savaient enfin ce qu'il y a au dos des fenêtres ». Dans le cadre du pliage permanent ou durable des documents, il s'agit d'une question pertinente. En effet, le pliage temporaire des fenêtres ne permet pas d'interagir avec le dos du document, puisque sitôt que l'utilisateur arrête l'interaction qui a déclenché le pliage pour commencer une interaction avec un autre élément (e.g. le dos du document), le document se déplie et le dos devient inaccessible. Avec le pliage durable des documents, les utilisateurs peuvent *interagir* avec le dos du document. Il nous semble naturel de considérer cette nouvelle zone comme le verso du document, de la même manière qu'une feuille de papier à un recto et un verso. Tout comme le verso d'une feuille de papier, le verso d'un document numérique sur une table interactive devrait être conçu de manière à être indistinguishable du recto du point de vue des interactions offertes. Vu que le recto de nos documents sur table interactive est composé d'éléments d'interface utilisateur (les *widgets*), le verso peut de façon similaire accueillir une interface utilisateur.

Nous avons implémenté, dans la librairie DiamondSpin, un système permettant de placer une interface utilisateur au dos des fenêtres. Par analogie avec le « *content pane* » (la zone principale d'une fenêtre) et le « *glass pane* » (une zone à fond transparent couvrant la fenêtre), nous avons appelé ce verso de la fenêtre « *back pane* ». Le *back pane* peut contenir une hiérarchie d'éléments d'interface utilisateurs de la même manière que le *content pane*. Ce contenu peut donc être récupéré dans une fenêtre d'une application existante, si elle a, à peu de choses près, la même taille que le recto. Au final, l'utilisateur peut interagir avec ces éléments de la même manière qu'avec ceux du devant du document, dans la limite, bien entendu, de la partie de *back pane* rendue visible par le pliage (figure 30).

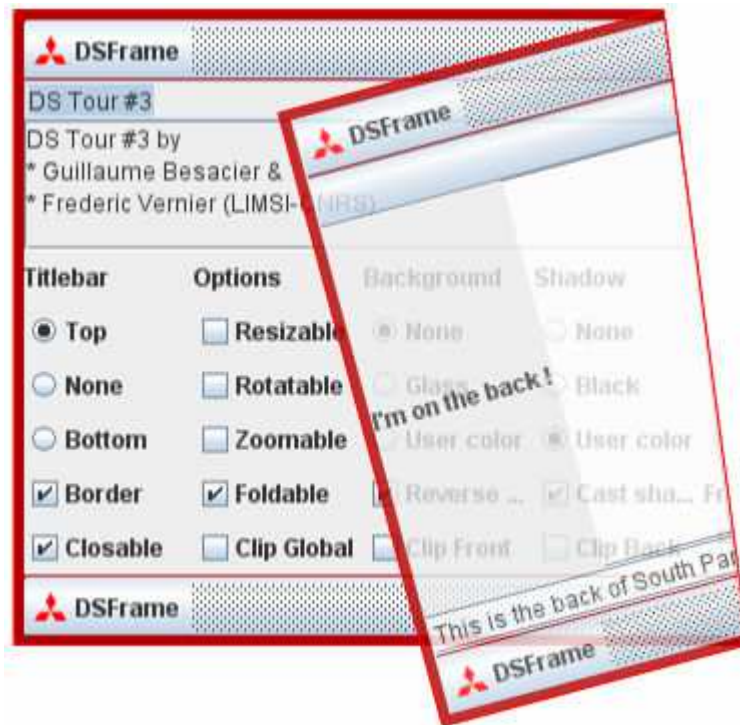


Figure 30 : Une fenêtre pliée. Le dos de fenêtre (back pane) contient des éléments d'une interface, avec lesquels l'utilisateur peut interagir.

3.4.1. Activation du pliage durable

L'interaction de pliage d'un document peut être déclenchée de différentes manières. Nous pensons que l'interaction de pliage doit être à la fois naturelle et évidente. Elle doit être évidente parce que c'est une nouvelle interaction pour un nouveau type de système interactif, les tables interactives. Si le pliage est déclenché par une option dans un sous sous-menu ou par un geste complexe avec le doigt, les utilisateurs devront apprendre à se servir du pliage plutôt que de découvrir le pliage. Avec une feuille de papier, il n'y a pas besoin d'apprendre à la plier. Une feuille de papier peut être pliée de manière évidente. Des pliages plus poussés, comme l'origami, ou le pliage d'une lettre façon « administrative », en trois tiers parfaitement égaux, peuvent en revanche requérir un apprentissage. Nous présenterons par la suite une interaction de cette nature basée sur le pliage. L'interaction de pliage doit également être naturelle parce que c'est une interaction générique qui peut être utilisée sur tous les documents présents sur la surface de la table interactive. L'interaction de pliage ne doit pas distraire ou gêner l'utilisateur lorsqu'il travaille sur ses documents.

Nous allons détailler trois façons de plier un document de manière durable, c'est-à-dire non temporaire, sur une surface interactive. Pour chaque technique, nous identifions un point mobile, qui est le point de référence du système de coordonnées local de la zone pliée du document. Le point mobile correspond souvent, mais pas toujours, à l'emplacement du doigt de l'utilisateur. Nous identifions également un point fixe, qui le point de référence de la position de la zone non pliée du document dans le système de coordonnées global de la table. Ce point fixe ne change pas de

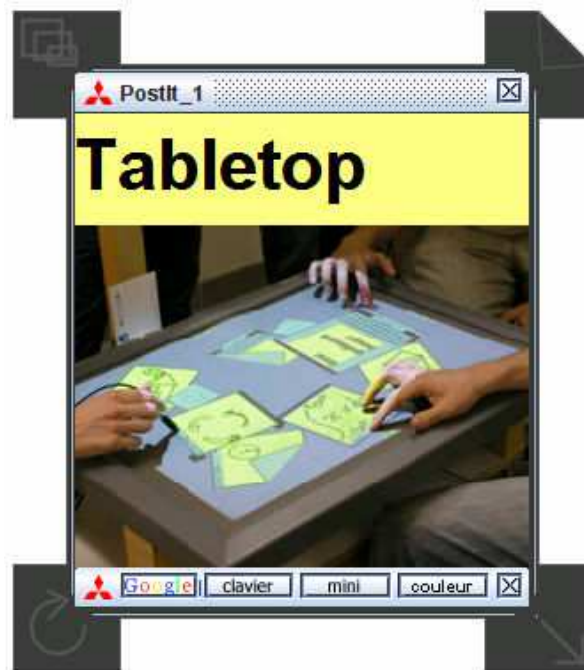


Figure 31 : Une fenêtre, représentant un post-it virtuel, créée avec la boîte à outils DiamondSpin. Les nouvelles interactions avec les fenêtres utilisent des interacteurs placés à l'extérieur de la fenêtre.

coordonnées pendant l'interaction, autrement dit la zone non pliée du document est immobile par rapport à la surface de la table.

Nous implémentons nos techniques d'interaction dans le cadre de la boîte à outils pour table interactive DiamondSpin. La première technique que nous proposons pour plier un document s'inspire des techniques d'interaction similaires déjà implémentées dans DiamondSpin. DiamondSpin permet en effet de nombreuses formes d'interaction (redimensionner, zoomer, tourner, etc. les documents). Ces interactions sont génériques et peuvent être utilisées sur tous les documents présents sur la surface de la table interactive. Les fenêtres créées avec la boîte à outils DiamondSpin incluent des interfaces utilisateur similaires à celles des ordinateurs de bureau. Elles sont composées de widgets tels que des boutons, des zones de textes, etc. modifiés pour être adaptés aux conditions d'une table interactive. Les fenêtres DiamondSpin peuvent également contenir l'interface utilisateur d'applications existantes. C'est un point important à prendre en compte dans la conception de nouvelles interactions avec les fenêtres. La plupart des prototypes de nouvelles interactions considèrent les fenêtres comme des zones vides, e.g. [6] propose des interactions pour tourner et plier les fenêtres en cliquant à l'intérieur de celles-ci. Or, l'intérieur des fenêtres peut contenir une interface utilisateur. Il s'agit du focus de cette thèse. La manipulation de photographies ou de documents statiques, aussi intéressante qu'elle soit, ne pose pas ces problèmes. Elle a par ailleurs été traitée extensivement dans la littérature existante. Avec des fenêtres qui contiennent une interface utilisateur, si l'utilisateur clique à l'intérieur de la fenêtre, il va activer un composant de cette interface. L'approche que nous avons suivie avec DiamondSpin

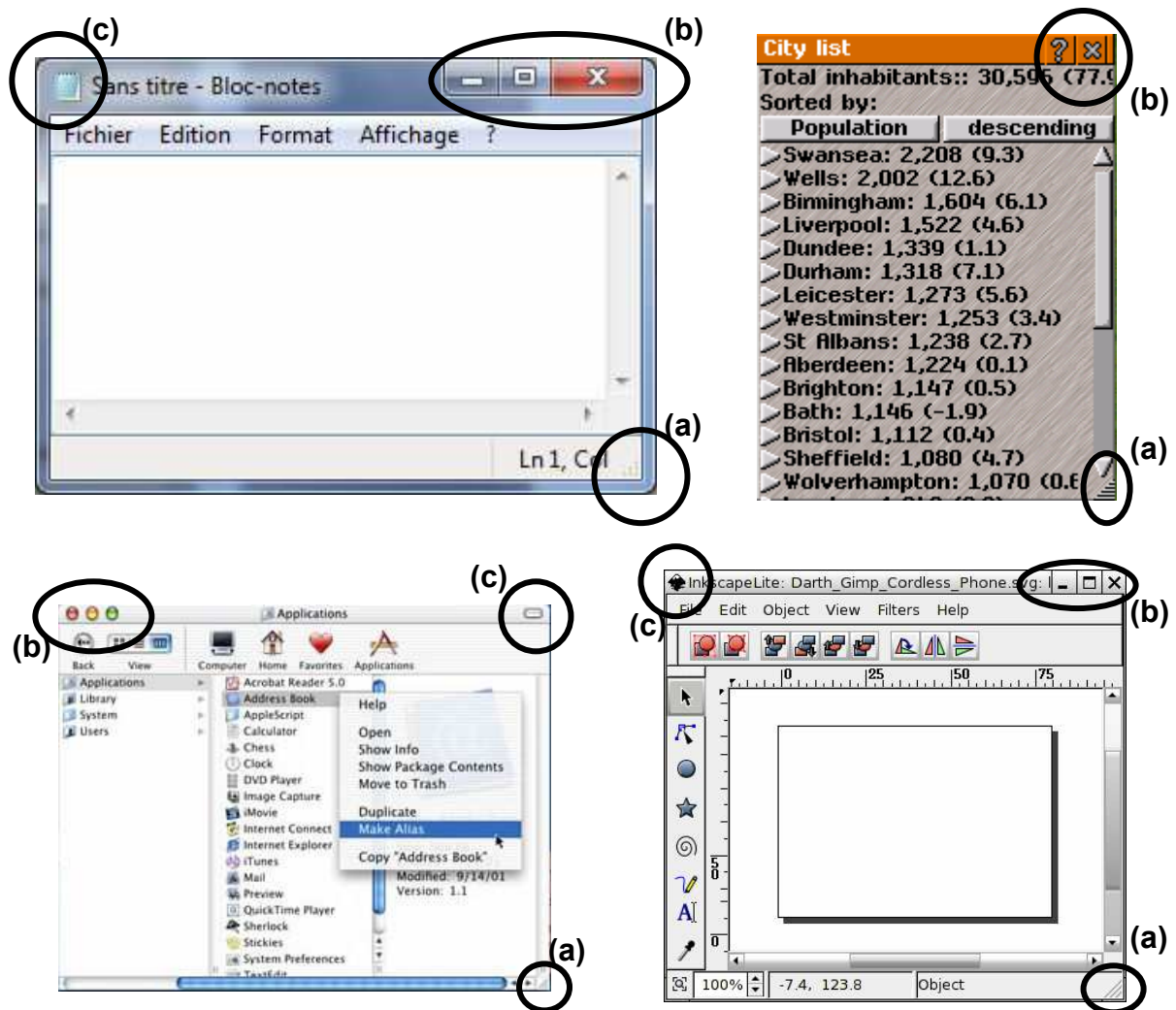


Figure 32 : L'utilisation des coins des fenêtres dans les environnements pour ordinateurs de bureau : (a) poignée de redimensionnement, (b) boutons de gestion de la fenêtre, (c) menu système.

est d'utiliser l'extérieur de la fenêtre pour activer les nouvelles interactions (figure 31).

Plusieurs interacteurs sont placés près des coins de la fenêtre. Les coins de la bordure des fenêtres sont une zone privilégiée pour le placement d'interacteurs dans les environnements d'ordinateurs de bureau (figure 32). L'extérieur de la fenêtre n'interfère pas avec les interacteurs existants dans l'application (sur la figure 31, une barre de titre et un bouton de fermeture sur la bordure du haut et une barre d'outils sur la bordure du bas). Il offre plus de place que l'intérieur de la fenêtre, ce qui permet une interaction digitale (i.e. au doigt) plus facile. L'aspect visuel des interacteurs répond à notre besoin de découvrabilité en offrant une indication visible que cette partie de la table déclenche une interaction. Pour ne pas surcharger visuellement la surface de la table interactive, qui peut contenir de nombreux

documents, ces interacteurs ne sont affichés que sur les fenêtres qui sont le focus d'interaction d'un des utilisateurs.

Dans DiamondSpin, le coin de pliage est celui situé en haut à droite. Les utilisateurs peuvent saisir un coin du document et le ramener vers l'intérieur de celui-ci pour plier le document sur lui-même. Le recto du document demeure immobile tandis que le verso du document suit le doigt de l'utilisateur (figure 36). Le système de coordonnées local de la zone pliée du document est basé sur ce point mobile, qui correspond à l'emplacement du doigt de l'utilisateur, et est donc toujours le coin supérieur droit du document. Le point de référence de la zone non pliée du document est le coin opposé. Même quand le document est plié au maximum, ce point reste sur le recto du document. Le choisir comme référence de la position de la zone non pliée du document dans le système de coordonnées global de la table permet de simplifier l'algorithme de pliage et d'affichage du document plié.

L'utilisation exclusive du coin supérieur droit pour l'interaction de pliage introduit des contraintes sur les types de pliages qui peuvent être obtenus. Nous avons choisi le coin supérieur droit après avoir observé qu'il s'agit du coin qui offre le plus de possibilité pour la création de techniques d'interaction métaphorique (e.g. tourner une page). Cette observation est confirmée par de nombreux exemples de conception de représentations graphiques des documents : un rectangle dont le coin supérieur droit est retourné (figure 33). Nous avons également donné cette forme standard à l'interacteur à saisir pour plier le document (figure 31, en haut à droite).



Figure 33 : Diverses icônes de documents, utilisées par des environnements graphiques aussi divers que des systèmes d'exploitation pour ordinateurs de bureau (Windows, Mac OS X, Linux) et des projets d'environnements interactifs innovants. Leur point commun est que toutes les icônes présentées ont leur coin supérieur droit plié.

La deuxième façon de plier une fenêtre sur une table interactive que nous avons envisagée prend sa source dans l'interaction gestuelle. Elle consiste à effectuer un mouvement rapide du doigt du bord de la fenêtre vers l'intérieur de la fenêtre. Avec cette technique, le document peut être plié de façon plus libre. L'emplacement sur la bordure du document où l'utilisateur commence le geste de pliage du document devient le point de référence du système de coordonnées local de la zone pliée du document. Il se déplace avec le doigt de l'utilisateur. Comme pour la technique précédente, la zone non pliée du document est immobile par rapport à la surface de la table. Nous utilisons également le même algorithme de pliage que pour la technique

précédente, avec comme point de référence de la position de la zone non pliée du document le point diamétralement opposé à l'origine du pliage (voir figure 36).

Par rapport à la technique précédente, cette technique de pliage de document nous semble être un moins bon compromis entre la nécessité d'une interaction métaphorique et les besoins d'une interface utilisateur utilisable. Elle permet de plier le document plus librement : l'utilisateur peut obtenir n'importe quelle forme de pliage qu'il pourrait souhaiter. Elle peut également supporter plusieurs pliages successifs du même document. Ce dernier point nous semble avoir un intérêt limité : l'affichage en deux dimensions du document plié ne permet pas de savoir précisément quelle face est au dessus de quelle autre dans une configuration complexe. Les interactions que nous proposons ne permettent pas non plus de spécifier si l'on plie le document vers l'avant ou vers l'arrière pour la même raison.

Cette liberté de pliage a en revanche un coût en utilisabilité qui nous semble plus élevé que l'avantage qu'elle apporte. L'utilisation d'un geste, sur une zone existante, non marquée, de la fenêtre, n'est pas aussi découvrable qu'un interacteur ayant une représentation visuelle. Plusieurs utilisateurs ont commenté, en essayant la démonstration de *Fold'n'drop* [25] (voir figure 29 en pages précédentes), qu'ils ne voyaient aucun changement par rapport au système habituel. Ces utilisateurs n'avaient simplement pas trouvé le geste qui permet de déclencher l'interaction de pliage. Pourtant, *Fold'n'drop* inclut un indice visuel sous la forme d'une courte animation lorsque l'utilisateur passe le curseur de la souris au dessus de la zone où il peut faire le geste de pliage. Dans notre cas, ce type de feedback n'est pas possible sur une table interactive où il n'y a pas d'événements produits quand l'utilisateur survole la table avec son doigt sans la toucher.

L'utilisation d'un geste sur un élément de l'interface utilisateur aussi commun que les bordures d'une fenêtre pose également un problème de déclenchement accidentel. Pour supporter l'interaction au doigt, qui est relativement peu précise, la taille de la zone qui réagit aux gestes doit être plus grande que l'élément graphique représentant la bordure de la fenêtre. Sur une table interactive de taille modeste (telle que les tables « grande DiamondTouch » (107 cm de diagonale) que nous utilisons), dès qu'un certain nombre de fenêtres sont présentes sur la table, une grande partie de la surface de travail devient sensible au déclenchement accidentel d'interactions basées sur les gestes. Ceci est contraire à notre but de ne pas distraire ou gêner l'utilisateur lorsqu'il travaille sur ses documents.



Figure 34 : L'interface graphique *BumpTop* utilise un stylet et un moteur de simulation des règles de la physique pour interagir avec les icônes [2].

Enfin, la troisième façon que nous avons envisagée pour plier les documents est d'utiliser un moteur de simulation de réaction physique. L'utilisation de règles basées sur le monde physique permet de créer des techniques d'interactions plus naturelles. Dans *BumpTop* [2] (figure 34), une interface utilisable au stylet, le déplacement d'icônes de documents est géré par un moteur de simulation des règles de la physique (collision, inertie, gravité,...). L'utilisation d'interactions utilisant ce moteur, telles que le jeter ou l'empilement, a permis une découvrabilité plus importante. 88% des tâches ont pu être complétées sans entraînement auparavant. Les utilisateurs ont également trouvé que le système leur semblait familier, y compris parmi les participants qui ne sont pas habitués à utiliser des environnements d'ordinateurs de bureau.

Une table interactive nous semble être un support approprié pour l'utilisation de telles interactions. En effet, la table elle-même a une existence physique. En particulier, la table délimite une surface de travail finie. Autour d'une table non-interactive, des utilisateurs travaillant avec des documents papier savent que les documents sortant des limites de la table vont tomber sur le sol. Notre technique d'interaction utilise cette connaissance intuitive du comportement du papier et des règles de la physique.

Les tables interactives que nous utilisons ont une bordure relevée relativement abrupte tout autour de la surface tactile. Nous avons étudié une interaction de collision entre cette bordure, physique, et les documents, numériques, affichés sur la table. Quand un document rencontre la bordure, il se plie lors de la collision (figure 36). Par extension, nous avons ajouté la possibilité d'autres obstacles virtuels (« murs », autres documents appartenant au même utilisateur ou à un autre utilisateur de la table,...) sur la table que les documents ne peuvent pas franchir. Une

interaction de jeter, avec un moteur de simulation de l'inertie, permet également de plier complètement un document en le projetant en direction d'un obstacle.

Cette technique a l'avantage de ne pas introduire de nouvelles interactions avec le document. Elle utilise l'interaction de déplacement à laquelle les utilisateurs sont habitués. Elle est facilement découvrable, spécialement avec la taille relativement réduite des tables interactives : les utilisateurs vont forcément heurter un bord de la table en déplaçant un document. Les études sur l'utilisation de la surface d'une table par plusieurs utilisateurs [103] indiquent que la zone la plus près du bord de la table est une zone privilégiée pour l'interaction (figure 35). Dans l'étude de deux ou trois utilisateurs collaborant à la création d'un plan sur une table circulaire, plus de 40% des interactions ont eu lieu la périphérie de la table (définie comme la zone entre le bord de la table et un cercle de rayon 75% de celui de la table), et seul 33% ont eu lieu au milieu de la table (défini comme un cercle de rayon 50% de celui de la table).

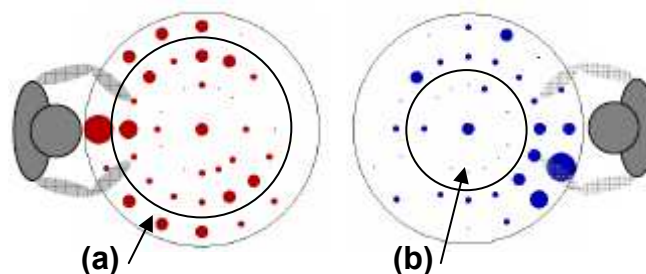


Figure 35 : Le bord de la table (zone a) est une zone privilégiée pour l'interaction, par rapport au centre de la table (zone b). La taille des points dénote le nombre d'interactions avec cette zone de la table [103].

Cette technique minimise également les désagréments d'un déclenchement involontaire du pliage d'un document. L'utilisateur ne peut pas interagir avec un document situé en dehors de la surface tactile. Le fait que le document soit plié lorsqu'il sort de cette surface ne réduit donc pas les possibilités d'interaction qu'a l'utilisateur avec le document. Au contraire, en permettant à l'ensemble du document de rester sur la surface tactile, et en découvrant une zone précédemment masquée du document, cette technique peut augmenter le nombre d'interactions possibles avec le document (par exemple en affichant au dos du document une interface pour sauvegarder le document, et ainsi éviter de le perdre avant qu'il ne sorte complètement de la table).

En revanche, il peut être complexe pour l'utilisateur d'obtenir exactement la forme de pliage qu'il veut sur un document. En effet, avec cette technique d'interaction, le point de référence du système de coordonnées local de la zone pliée du document ne correspond pas à l'emplacement du doigt de l'utilisateur, contrairement aux deux précédentes techniques. A la place, il s'agit du point miroir de celui qui a touché en premier l'obstacle à éviter. Contrairement aux deux précédentes techniques aussi, la zone non pliée du document ne reste pas immobile sur la table. Il n'y a donc pas de point de référence de la position de la zone non pliée du document dans le système de

coordonnées global de la table. Un algorithme de pliage différent de celui des deux autres techniques est mis en œuvre pour cette technique.

La figure 36, ci-dessous, résume les trois façons de plier un document que nous avons présentées.

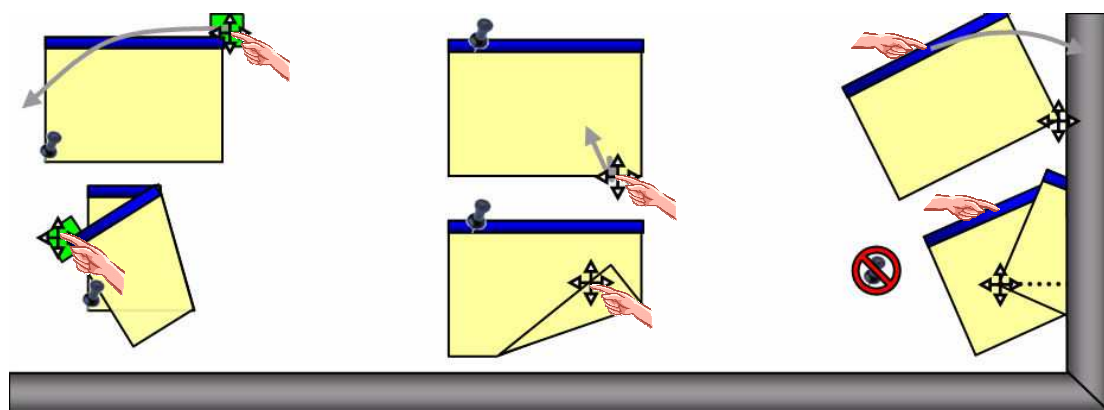


Figure 36 : Trois façons de plier un document (de gauche à droite : avec un interacteur dédié, avec un geste du doigt, avec un moteur physique). Pour chaque technique, la ligne grise indique le mouvement de l'utilisateur, les quatre flèches indiquent le déplacement du verso du document, et la punaise indique la position du point fixe sur le recto du document.

3.4.2. Interaction avec un document plié

Le pliage durable que nous proposons conserve tous les avantages associés au pliage temporaire qui ont été identifiés dans les environnements pour ordinateurs de bureau. Il peut être utilisé pour révéler les documents présents sous le document plié. L'aspect durable du pliage permet d'interagir avec ces documents sans déplacer ni le document, plié, qui gênait cette interaction, ni les documents découverts par le pliage. Si la position spatiale des documents sur la table représente une information (par exemple des documents disposés de gauche à droite chronologiquement sur la table), ne pas avoir besoin de déplacer les documents permet de préserver cette information.

Une conséquence visuelle immédiate du pliage d'un document est de faire apparaître un nouvel artefact numérique : le dos du document. Le pliage durable offre à l'utilisateur de nouvelles façons d'interagir avec le document pendant que celui-ci reste plié. Nous avons implémenté, dans la librairie DiamondSpin, un système permettant de placer une interface utilisateur au dos du document, le « back pane ». L'utilisateur peut interagir avec cette interface de la même manière qu'avec le devant du document, dans la limite, bien entendu, de la partie de back pane rendue visible par le pliage.

(recto)

(verso)

Figure 37 : Des formulaires papier de l'administration française. Le verso des formulaires contient des indications sur la façon de remplir le recto, ou plus généralement des informations sur le processus dont le formulaire fait partie.

Dans le cadre de la métaphore du papier, le dos du document présente naturellement des propriétés similaires au verso d'une feuille de papier. Ce verso du document nous semble particulièrement adapté à certains usages. A nouveau, nous avons observé des documents papier existants pour proposer une interaction la plus métaphorique possible. Ainsi, les formulaires contiennent souvent au verso des explications ou des conseils sur la façon de remplir le recto (voir par exemple la figure 37, premier formulaire). Cette pratique est suffisamment courante pour des utilisateurs qui sont présentés un nouveau formulaire retournent la page et vérifient la présence d'un verso avant même d'avoir fini de lire le recto. Le verso de tels formulaires peut également indiquer l'adresse où envoyer le formulaire, la liste des pièces à joindre, ou expliquer le processus dont fait partie le formulaire.

De manière similaire, nous proposons d'utiliser le verso de documents numériques pour afficher l'aide en ligne de l'application correspondante. La capacité d'une application à fournir de l'aide sur son fonctionnement à ses utilisateurs permet d'augmenter l'efficacité des utilisateurs [60]. Les systèmes d'aide conçus pour les ordinateurs de bureau sont souvent standardisés, par exemple l'utilisation de la touche du clavier F1 pour appeler l'aide, la présence de rubriques, la recherche par mots-clefs. Pourtant, ils souffrent de plusieurs problèmes connus [26], tels qu'une charge cognitive trop élevée, des problèmes d'accès simultané à l'aide et à l'application, et une perte d'efficacité due à l'interruption de la tâche.

L'approche métaphorique que nous proposons se base sur les connaissances existantes de l'utilisateur, sur son affinité avec les objets du monde réel, pour réduire la charge cognitive liée à l'utilisation du nouveau système informatique. L'utilisateur peut simplement retourner une fenêtre de l'application pour lire, au verso de celle-ci, de l'aide à propos des fonctionnalités du recto.

L'utilisation de l'informatique permet toutefois d'ajouter un peu de la « magie » que nous évoquons dans l'introduction de ce chapitre. L'aide affichée au dos du document peut s'adapter à l'état d'interaction du document. L'application peut ne pas afficher d'aide pour les parties déjà correctement remplies de l'interface utilisateur. Ceci permet de diminuer le temps que l'utilisateur passe à chercher l'information dont il a besoin dans l'aide. Dans la figure 37, l'aide au verso du premier formulaire est organisée en sections numérotées qui correspondent aux cadres numérotés du recto du formulaire. La version numérique sur table interactive de ce formulaire n'afficherait pas les sections d'aide concernant les cadres déjà remplis. Le système d'aide peut également s'initialiser pour afficher en premier l'aide concernant le composant de l'interface utilisateur actuellement sélectionné à la place du sommaire de l'aide.

Les formulaires papier contiennent aussi, au dos, des informations sur le processus dont le formulaire fait partie. Dans le cadre d'une application informatique, le cycle de vie d'un document utilise des opérations comme la sauvegarde sur disque dur, la gestion de versions, l'impression papier du document, etc. Dans le cadre de notre métaphore papier pour table interactive, ces opérations peuvent être accessibles via le

verso du document, comme dans la figure 38. L'application de dessin présentée dans cette figure permet de créer des « feuilles de dessins » virtuelles sur la table interactive. Chaque feuille affiche sur son recto une palette d'outils de dessin (trait, remplissage, couleur, gomme). Cette palette permet de créer et modifier le contenu de la fenêtre : le dessin. La gestion de la feuille de dessin se fait à l'aide d'outils placés au dos de celle-ci : sauvegarder, copier, imprimer,... Cette seconde interface permet de gérer le support du dessin, la « feuille à dessin » elle-même. Les applications pour ordinateur de bureau mélangent communément ces deux types d'outils (figure 39). Pour les documents physiques, en revanche, l'usage qui en fait sépare ces deux types d'actions temporellement, et parfois physiquement.

Dans le cadre de l'utilisation d'applications pour ordinateurs de bureau existantes, certaines des technologies décrites au chapitre premier permettent d'obtenir la liste des telles actions. Ainsi la preuve de concept concernant les menus d'une application permet de récupérer des pointeurs sur ces actions grâce aux menus standardisés des applications. La technologie utilisant les langages de script, quand à elle, permet de lancer de telles actions directement sur les données d'une application. Ceci nous permet d'utiliser des techniques d'interaction innovantes, comme celles que nous présentons ici, avec des applications pour ordinateurs de bureau existantes.



Figure 38 : Une application pour table interactive de dessin. Le recto de l'application présente une interface minimale contenant uniquement les outils de dessin (trait, remplissage, couleur, gomme). Le verso de chaque document de dessin contient une interface pour gérer le document au recto (sauvegarder, copier, imprimer,...).

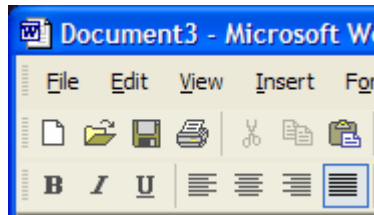


Figure 39 : Les applications pour ordinateurs de bureau (ici Microsoft Word) affichent communément les outils pour créer et modifier le contenu du document au même plan que les outils pour gérer le document lui-même.

L'interface de gestion au dos du document peut être identique pour plusieurs types de documents. Les actions qu'elle propose à l'utilisateur ne sont pas plus spécifiques à un document de dessin qu'à un document écrit ou à des types de documents spécifiques comme une liste d'inventaire, un document comptable, etc. Cette uniformité permet de diminuer le nombre d'interfaces différentes que l'utilisateur doit maîtriser.

Les actions qui peuvent être matérialisées au dos des fenêtres sont parfois accomplies, dans les applications pour ordinateurs de bureau, avec une suite de boîtes de dialogues. Sur une table interactive, le système de boîtes de dialogue peut devenir ambigu pour les utilisateurs. En effet, le lien entre la fenêtre principale d'un utilisateur, qui contient la représentation graphique du document, et les boîtes de dialogue, qui apparaissent pour effectuer une action sur ce document, n'est pas visuellement représenté. Sur les ordinateurs de bureau, un unique utilisateur travaille avec une unique application à la fois, affichée au premier plan. Les boîtes de dialogue affichées, non seulement concernent forcément le document au premier plan, mais ont été ouvertes par l'utilisateur de l'ordinateur. L'utilisateur, lorsqu'il ouvre une boîte de dialogue, va faire un lien mental entre le document sur lequel il travaille et ce qu'il veut accomplir via la boîte de dialogue (e.g. imprimer le document). Sur une table interactive, plusieurs utilisateurs peuvent travailler en même temps sur plusieurs documents. Seul l'utilisateur qui a ouvert une boîte de dialogue donnée peut faire le lien mental entre celle-ci et un des documents. L'absence de lien visuel entre une boîte de dialogue et le document qu'elle concerne ne permet pas aux autres utilisateurs de savoir si une boîte de dialogue donnée concerne un de leurs documents, par exemple un document qu'ils viennent de transférer d'un autre utilisateur. Si plusieurs boîtes de dialogue identiques sont affichées, e.g. plusieurs boîtes de dialogue « imprimer », ouvertes par plusieurs utilisateurs pour des documents différents, aucun indice visuel ne permet aux utilisateurs de savoir quelle boîte de dialogue est celle qu'ils ont ouverte. Le dos d'un document, en revanche, est physiquement lié au document associé. En affichant les outils qui utilisent habituellement des boîtes de dialogue au dos du document concerné, nous levons l'ambiguïté.

Le dos du document permet également de créer un système d'interface modale. Les boîtes de dialogue modales sont des boîtes de dialogue qui figent l'interaction avec le reste du système tant qu'elles sont affichées. Une boîte de dialogue peut être modale

au niveau d'une fenêtre, d'une application ou de l'ensemble de l'ordinateur. Sur une table interactive, les boîtes de dialogues modales sont difficilement envisageables. La surface de la table est partagée par plusieurs utilisateurs : une application ne devrait pas pouvoir bloquer l'ensemble de la table à cause de l'interaction d'un seul utilisateur. Une boîte de dialogue modale est également problématique au niveau d'un seul document. La boîte de dialogue et le document pouvant être séparé, un utilisateur pourrait se demander pourquoi un document ne réagit plus sans remarquer qu'une boîte de dialogue modale le concernant est ouverte à un autre endroit de la table. L'utilisation du dos du document permet d'accomplir le même effet qu'une boîte de dialogue modale. Quand un document plié en deux à plus de 50%, son recto n'est simplement plus accessible. Il est physiquement bloqué par le recto qui est replié dessus. Nous utilisons un tel système pour confirmer la fermeture d'une fenêtre. Quand l'utilisateur touche la croix de fermeture d'un document modifié, celui-ci se plie automatiquement à 60%. Le dos du document affiche alors trois boutons : détruire le document, effacer le document et recommencer ou annuler la fermeture (figure 40). Tant que l'utilisateur n'a pas fait un choix, il ne peut pas déplier le document, et donc interagir avec lui. Le document est déplié automatiquement si l'utilisateur sélectionne effacer ou annuler (il est détruit s'il sélectionne fermer).

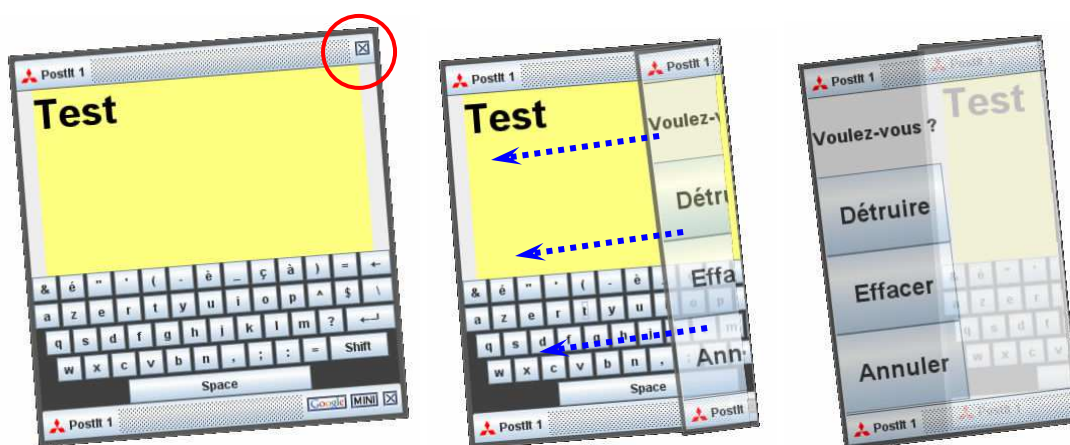


Figure 40 : Plier un document pour rendre physiquement inaccessible son recto permet de créer un système similaire aux boîtes de dialogues modales au niveau d'un document. Quand l'utilisateur sélectionne la croix de fermeture d'un post-it virtuel (à gauche), celui-ci se plie automatiquement à 60% et affiche une demande de confirmation au verso.

Pour les boîtes de dialogue modales au niveau de l'ensemble du système, nous pensons que les actions qu'elles déclenchent (par exemple, éteindre l'ordinateur-table interactive) doivent avoir l'assentiment de tous les utilisateurs. D'autres types d'interactions ont été proposés pour ceci. Nous apprécions tous particulièrement le système de gestes collaboratifs proposé par Morris *et al.* [77] (figure 41). Les utilisateurs se mettent d'accord entre eux, sans intervention du système informatique. Ils utilisent les protocoles sociaux de communication entre êtres humains, qui sont rendus possibles par la colocalisation des utilisateurs autour de la

table. Ils effectuent ensuite une interaction de façon synchronisée, ensemble, ce que le système informatique interprète comme un accord préalable entre les utilisateurs. Si un utilisateur n'est pas d'accord avec l'action, il peut continuer son travail sans interruption. Le système informatique reconnaîtra que tous les utilisateurs n'ont pas pris part à l'interaction et ignorera celle-ci.

Morris *et al.* proposent deux types de gestes collaboratifs pour ce but. Les utilisateurs peuvent faire la même action simultanément, chacun sur leur zone de la table. Par exemple, chaque utilisateur peut faire le geste d'effacement (frotter la table avec la paume de la main) sur sa zone, et le système l'interprétera comme une action d'effacement de toute la table. Cette technique est utilisable sur la plupart des systèmes de tables interactives. Le second type de gestes collaboratifs n'est utilisable qu'avec les tables DiamondTouch [24]. La table DiamondTouch peut détecter quand un utilisateur touche un autre utilisateur, grâce au contact électrique entre eux et les récepteurs sur lesquels ils sont assis. Les quatre utilisateurs se tiennent par la main pour former une chaîne de mains, puis l'un d'entre eux effectue une interaction. Le système détecte que les utilisateurs se touchaient lors de l'interaction, et l'interprète comme un accord préalable entre les utilisateurs.



Figure 41 : L'équivalent des boîtes de dialogues modales au niveau du système en entier peut utiliser des gestes coopératifs. A gauche, les quatre utilisateurs font le même geste d'effacement en même temps pour effacer tout le contenu de la table. A droite, les quatre utilisateurs forment une chaîne de mains, puis l'un d'entre eux appuie sur le bouton quitter pour éteindre l'ordinateur-table interactive [77].

Il est intéressant d'étudier les implications culturelles de cette interaction. Morris *et al.* notent qu'elle a reçue un accueil mitigé de la part de leurs utilisateurs [77], les utilisateurs qui étaient en couple n'ont pas objecté à se tenir la main, mais les utilisateurs qui étaient collègues ou amis ont trouvé le concept plus inhabituel. Nous utilisons fréquemment cette capacité des tables DiamondTouch lorsque nous faisons une démonstration de nos applications. En touchant un utilisateur, l'expérimentateur peut lui « voler » son identité du point de vue du système, et faire des interactions en son nom. Par exemple, quand un utilisateur n'arrive pas à comprendre le fonctionnement de la table interactive, nous mettons une main sur son épaule et effectuons l'interaction à sa place. Jusqu'à maintenant, aucun utilisateur n'a régit négativement.

D'autres types d'interfaces nous semblent pouvoir bénéficier de la mise en place d'une interaction avec le dos du document. Toujours dans l'idée de la métaphore du formulaire qui contient des méta-informations sur le processus dont il fait partie au recto du contenu primaire du formulaire, nous envisageons une application pour table interactive qui permette de consulter et d'éditer des ressources documentaires à l'aide de feuilles de papier virtuel affichant au dos de chaque document des informations sur la création, la fiabilité, le cycle de vie,... de la ressource. Les systèmes de gestion de contenu (CMS, *content managing systems*) proposent des systèmes collaboratifs, multi-utilisateur, distribués géographiquement, qui utilisent plusieurs interfaces graphiques séparées pour afficher ces informations. Ainsi, les applications web basées sur le système « wiki », tels que Wikipédia (figure 42), proposent actuellement ces informations sur plusieurs pages web distinctes.

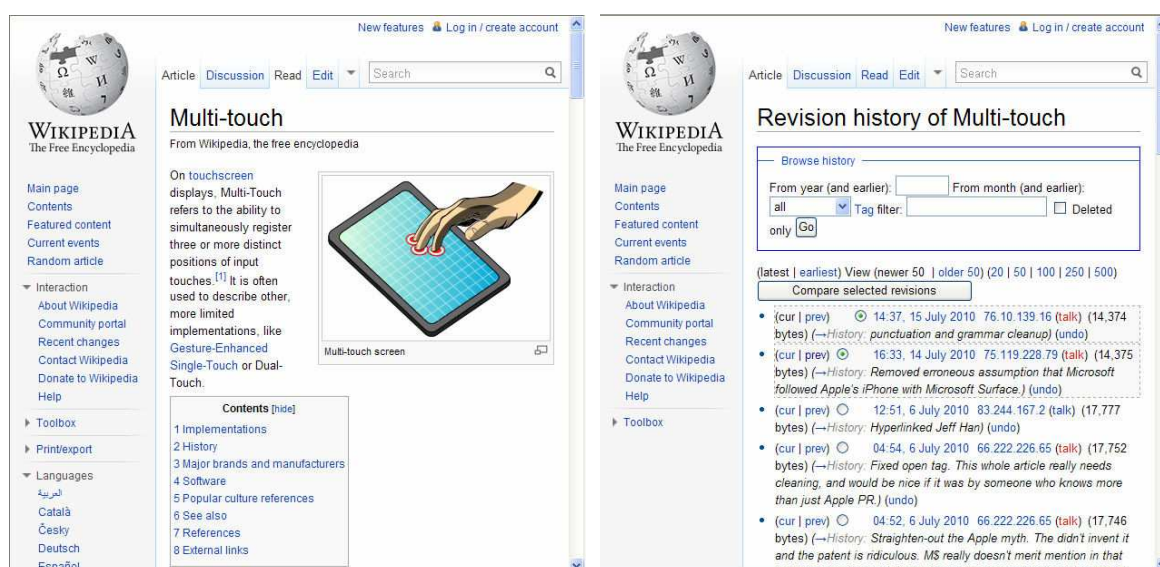


Figure 42 : Wikipédia utilise un système collaboratif de gestion de contenu de type « wiki », qui affiche le contenu et les méta-informations du contenu sur des pages distinctes.

Nous avons conçu un système de consultation et de partage de photographies sur une table interactive qui utilise le verso des images pour afficher des informations sur les sujets photographiés (figure 43). Les photographies papiers développées dans les laboratoires commerciaux sont marquées au dos d'un tampon indiquant la date de développement. Il est fréquent que les photographes ajoutent ensuite des informations comme le lieu de la prise de vue ou le nom des personnes photographiées à côté de cette date. Les photographies numériques sont utilisées par une table interactive sans avoir besoin d'être développées auparavant. Par un certain aspect, quand l'utilisateur connecte son appareil photographique numérique (APN) à la table interactive, nous pouvons dire que celle-ci « développe » les images contenues dans l'APN et les affiche sur la surface de la table. Nous prolongeons la métaphore du développement de photographies en affichant au dos des images des informations sur celles-ci. Ces informations peuvent provenir de plusieurs sources. Nous pensons par exemple aux données EXIF (*Exchangeable image file format* :

date, localisation GPS, type d'APN, etc.), à des données préalablement saisie par l'utilisateur, des données automatiquement extraites d'Internet (par exemple une recherche sur le web d'informations sur la localisation GPS), ou de reconnaissance automatique des visages dans la photographie.



Figure 43 : Le verso des photographies contient des informations sur la photographie et le sujet photographié. Une interface graphique utilisant des onglets permet de naviguer dans les informations.

Notre implémentation du pliage durable propose d'afficher le verso d'un document plié de façon légèrement transparente (figures 30, 38, 40, 43) pour permettre de voir le recto du document à travers le verso. Cela permet par exemple de reconnaître la nature du document. Si le verso du document contient une interface qui permet de modifier le recto du document, cela permet également de voir ces modifications au fur et à mesure qu'elles sont effectuées. Ce retour direct (*direct feedback*) du résultat des actions de l'utilisateur est similaire aux requêtes dynamiques [127]. La transparence d'un côté du document, mais pas de l'autre côté du même document ne correspond toutefois pas à la réalité physique du papier.

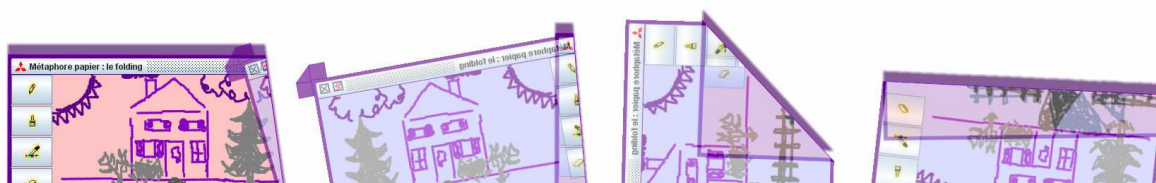


Figure 44 : Un document numérique composé de film de plastique virtuel. Les deux côtés du document sont identiques, à la couleur de fond près (le recto est en rose, le verso en bleu).

Nous considérons donc également le cas où le document serait composé non plus de papier virtuel, mais d'un film de plastique transparent virtuel. Dans ce cas, les deux côtés du document présenteraient le même contenu (figure 44). Retourner le document permet d'avoir une image miroir du contenu. Les documents, qu'ils soient

papiers ou transparents et réels ou virtuels, contenant du texte sont habituellement retournés selon un axe vertical, pour pouvoir lire le dos du document. En effet, s'ils sont retournés selon un autre axe, l'utilisateur ne voit pas le texte présent au dos du document dans le bon sens pour le lire. Pour un document tel qu'un dessin ou une photographie, en revanche, d'autres axes de rotation peuvent être utiles. Les documents de type films transparents, en particulier, permettent de voir le contenu du document sous un autre angle selon l'axe selon lequel ils sont retournés.

La figure 44 présente trois axes possibles (de gauche à droite) : le document original non retourné, l'axe vertical pour voir une copie miroir du document, l'axe incliné à 45° pour montrer une copie miroir à l'utilisateur assis à sa droite, et l'axe horizontal pour montrer une copie miroir à l'utilisateur assis en face de soi. Le pliage d'un document peut ainsi devenir une interaction qui fait intervenir plus d'un utilisateur. Nous le proposons comme une alternative rapide à la rotation du document, avec l'avantage non négligeable de pouvoir être annulé facilement. Il suffit de déplier le document pour qu'il retrouve exactement son orientation et sa position précédente. Avec la rotation, l'utilisateur ne peut pas retrouver facilement la même orientation. Certains problèmes techniques avec les technologies de table interactive actuelles (entre autres, l'utilisation de surface d'affichage carrée sujette aux problèmes l'aliasing lors de l'affichage d'objets non alignés avec les bordures de l'affichage) font préférer certaines valeurs pour l'orientation au degré près.

Il est intéressant de noter que si la technique d'interaction utilisée pour plier le document limite les types de pliages qui peuvent être obtenus (e.g. l'utilisation d'un coin de pliage précis, comme celui présenté sur la figure 31), cela introduit une dissymétrie entre les utilisateurs. Dans notre exemple, cela ne permet pas de montrer le verso du document au participant situé à gauche de l'utilisateur. Cette dissymétrie pourrait conduire les participants à ne pas se répartir aléatoirement autour de la table, mais à privilégier les configurations où ils ont le plus de possibilités techniques d'interactions avec le ou les participants avec lesquels ils doivent le plus interagir dans le cadre de leur tâche collaborative.

3.5. Plier pour empiler

La métaphore du papier nous a inspiré une technique d'interaction pour créer et gérer des groupes de documents sur une table interactive. En effet, sur une table interactive, l'espace physique de la surface sert de support aux documents eux-mêmes⁵. Un système d'organisation des documents en est d'autant plus nécessaire. Les piles ont été utilisées sur des ordinateurs de bureau [72], où elles permettent d'offrir un système de catégorisation moins rigide tout en étant plus reconnaissable. En effet, l'apparence extérieure d'une pile informe sur son contenu : les utilisateurs peuvent reconnaître les documents dans la pile et l'identifier, bien que celle-ci n'aie pas de nom. Les piles sont également plus facilement parcourables.

Dans BumpTop [2], une interface utilisable au stylet, les piles sont combinées à un moteur de simulation des règles de la physique (collision, inertie, gravité,...). Elles ont permis une découvrabilité plus importante, ainsi 88% des tâches ont pu être complétées sans entraînement auparavant. Les utilisateurs ont également trouvé que le système leur semblait familier, y compris parmi les participants qui ne sont pas habitués à utiliser des environnements d'ordinateurs de bureau.

Nous avons noté, dans ces différentes implémentations, une grande variété dans la forme des piles et les façons de les parcourir. Par exemple, dans certaines piles les documents sont parfaitement empilés et leurs bords alignés, dans d'autres les documents sont éparpillés sur une plus grande surface, d'autres encore imposent des positions fixes aux documents comme un éventail ou une grille de documents.

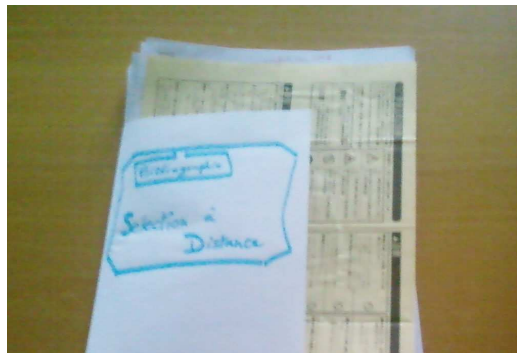


Figure 45 : Une pile de papiers maintenus par une feuille de papier pliée en deux. La feuille de papier pliée fait office de couverture de la pile, et peut être annotée.

⁵ Dans les environnements pour ordinateurs de bureau actuels, les surfaces (comme le bureau métaphorique) servent de support à des icônes de documents. Les icônes, une représentation abstraite et standardisée (e.g. en taille), sont plus faciles à organiser (e.g. sur modèle de grille).

Avec des documents papiers, nous avons pu observer des ensembles de documents maintenus par une feuille de papier pliée en deux en guise de couverture (figure 45). Cela nous a mené à imaginer une utilisation supplémentaire à la technique de pliage. Les documents pliés sur la table interactive conservent les avantages des documents papier pliés : on peut plier un document pour voir les documents qui sont dessous, ou pour jeter un coup d'œil à son verso. Il semble donc tout naturel de pouvoir s'en servir pour faire des piles. Ainsi, en utilisant le pliage pour résoudre le problème de l'empilement, nous proposons une interaction de plier pour empiler (ou *Peeling for Piling*). Le point principal est d'ouvrir la possibilité de créer des piles de documents de façon opportuniste, sans devoir le planifier à l'avance, et naturelle, sans devoir faire intervenir une interface ou des interacteurs supplémentaires sur la table interactive.

La propriété principale des piles est de pouvoir être manipulée ensemble. Ainsi, l'ensemble de la pile de documents papiers peut être déplacée en une seule fois. La pile se comporte de ce point de vue comme un document unique. Sur une table interactive, nos piles numériques (figure 46) reprennent cette propriété. La pile, c'est-à-dire le document plié de couverture et tous les documents présents dans la pile, peut être déplacée en utilisant la même interaction que celle qui est utilisée pour déplacer un simple document. Nous étendons cette propriété aux interactions de rotation et de zoom qui sont disponibles pour un document dans notre environnement de table interactive. Plier une pile, en revanche, ne nous semble pas pertinent. Le pliage d'une pile impliquerait que sa couverture serait pliée deux fois : une fois avant de créer la pile, pour servir de couverture, et une seconde fois en même temps que le reste de la pile. Comme nous l'avons évoquée plus haut, plier plus d'une fois un même document nous semble introduire une complexité contre-productive dans la métaphore du pliage. L'affichage en deux dimensions du document plié ne permet pas de savoir précisément quelle face est au dessus de quelle autre, et les interactions que nous proposons ne permettent de spécifier si le document est plié vers l'avant ou vers l'arrière.

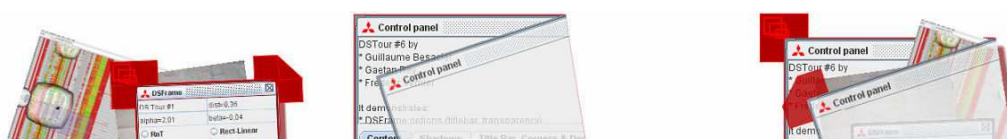


Figure 46 : Avant et après : création d'une pile numérique. Les documents (emploi du temps, tableur, globe terrestre, boîte de dialogue) sont glissés entre les deux faces d'un document plié en deux. L'ensemble forme d'une pile.

En groupant les documents, les piles numériques permettent entre autres à plusieurs utilisateurs d'échanger un ensemble de documents plus facilement. Les documents peuvent être déplacés en une seule interaction sur la surface de la table. De même, l'utilisateur qui reçoit les documents peut les orienter pour son sens de lecture en une seule rotation, puis les sortir un par un de la pile pour les manipuler. Au contraire, des documents non groupés devraient être déplacés un par un, ce qui peut être très pénible si la distance de déplacement est plus grande que la portée du bras de l'utilisateur. Ce dernier devrait systématiquement se lever de son siège pour déplacer un document jusqu'au bout, puis revenir à sa place pour prendre le document suivant, et se redéplacer à nouveau. Ce type de routine nous évoque plus une parodie burlesque qu'un environnement de travail productif. Une fois tous les documents déplacés, il resterait encore à tous les orienter dans la direction de l'utilisateur qui va les utiliser.

Grouper des documents ensemble permet également d'utiliser la surface de la table pour stocker plus de documents. Dans le cas de documents papier, ce gain d'espace est réalisé par l'utilisation d'une troisième dimension pour le stockage de document : la hauteur. Les documents sont physiquement les uns par-dessus les autres, donc ils prennent moins d'espace dans les deux dimensions du plan horizontal. Ceci est également vrai dans le cas de piles numériques. Toutefois, en permettant de contrôler le zoom de l'ensemble de la pile, notre interaction de plier pour empiler permet de réduire encore plus l'espace nécessaire sur la table interactive pour stocker les documents. A nouveau, la capacité de contrôler l'ensemble des documents de la pile en une seule interaction facilite la gestion des documents. Un utilisateur peut, par exemple, rétrécir une pile de document dans un coin de la table. Les travaux de Mander *et al.* [72] montrent que la pile, en permettant de voir des morceaux de documents dépasser de la couverture, reste reconnaissable. Quand il recherche un document, l'utilisateur peut reconnaître la pile miniaturisée qui le contient, agrandir l'ensemble de la pile pour pouvoir la parcourir, extraire son document et re-miniaturiser la pile.

Les piles numériques ont un avantage majeur par rapport aux piles papier : elles ne sont pas assujetties aux lois de la physique. Elles peuvent faire usage de cette magie dont parle Alan Kay, et que nous avons évoqué dans la présentation de la métaphore papier. Dans le monde physique, le document utilisé comme couverture d'une pile est parfois annoté, comme sur la figure 45. Parmi les annotations que nous avons pu voir se trouve la date de dernière mise à jour de la pile, le nombre de documents dans la pile ou une liste de documents manquant encore dans la pile (par exemple une liste de pièces à réunir pour joindre à un dossier) barrée au fur et à mesure que les documents sont ajoutés. L'espace au dos de la couverture du document numérique qui maintient, métaphoriquement, ensemble les documents d'une pile numérique peut être utilisé pour afficher de telles informations sur la pile. En revanche, le système informatique peut être conçu pour que ces annotations se mettent à jour automatiquement lorsque l'utilisateur manipule la pile. De manière similaire, dans une pile papier, le document qui a été recyclé en couverture doit avoir un côté (en

général le verso) blanc pour permettre ces annotations. Dans le cas numérique, le document qui a été plié en deux a également perdu sa fonction primaire de document. Son recto n'est plus accessible, voir masqué par les documents contenus dans la pile. Il a été « recyclé » en couverture de la pile. Si son recto contenait une interface utilisateur, celle-ci n'est plus nécessaire. Elle peut être enlevée et remplacé par l'espace pour les annotations sur le contenu de la pile. L'utilisateur peut donc utiliser n'importe quel document comme couverture, sans avoir à se soucier de savoir si le verso est blanc ou non.



Figure 47 : Une interface peut être placée au dos de la couverture de la pile pour contrôler l'ensemble de la pile.

Ces annotations peuvent être étendues en une interface utilisateur pour contrôler l'ensemble de la pile (figure 47), par exemple trier les documents à l'intérieur de la pile, afficher une mosaïque des documents contenus dans la pile, envoyer la pile par e-mail, etc.

Une pile de documents d'un type spécifique peut avoir une signification supplémentaire. Par exemple, dans le monde physique, chaque transparent pour rétroprojecteur est un document séparé. Une pile ordonnée de ces transparents est une présentation. Les logiciels de présentation pour ordinateurs de bureau, tel que Microsoft PowerPoint, ne fonctionnent pas de cette façon. Dans ces logiciels, un transparent n'existe pas en dehors d'une présentation, cette dernière étant le document. L'interface graphique de ces logiciels ne permet, par exemple, d'éditer deux transparents en même temps. Une application collaborative pour table interactive, au contraire, pourrait se baser sur une métaphore de transparents individuels, qui sont ensuite réunis dans une pile. L'interface de contrôle au dos de la couverture de pile peut s'adapter au type de documents que la pile contient. Pour une pile de transparents, cette interface peut permettre de lancer la présentation sur un vidéo-projecteur, d'exporter l'ensemble de la pile dans un fichier compatible avec des logiciels comme PowerPoint, etc.

Ce mécanisme peut être étendu à divers types de documents. Les bases métaphoriques utilisées dans la conception de cette technique d'interaction permettent à l'utilisateur de transférer ses connaissances des interactions du monde réel aux interactions sur une table interactive. Nous proposons l'ajout de « magie » à la pile. La pile, en plus d'une façon simple de garder ensemble des documents, devient un outil pour interagir avec un ensemble de document, et forme elle-même un document.

Parmi les autres types de documents que nous envisageons, nous pouvons citer des piles de cartes de visites virtuelles permettant d'envoyer un message à tous les contacts présents dans la pile. Une telle pile peut également prendre une forme distinctive pour rendre les interactions supplémentaires plus visibles. Une pile de cartes de visite peut prendre la forme d'un rolodex (figure 48), avec un ensemble d'onglet pour accéder aux cartes commençant par la lettre correspondante.



Figure 48 : Un rolodex est un type de carnet d'adresses se présentant sous la forme d'un ensemble ordonné de cartes intercalées d'onglets pour accéder rapidement à une carte donnée.

3.5.1. Activation du plier pour empiler

Sur la figure 46, présentant la situation avant et après la création d'une pile, nous n'avons que suggéré les actions qui se déroulaient entre les deux étapes. La situation avant l'empilement nous semble claire : plusieurs documents indépendants sont disposés sur la table interactive. La situation après l'empilement est tout aussi claire, grâce à son aspect métaphorique : la couverture de la pile est pliée en deux, les autres documents sont dedans, l'ensemble se manipule comme s'il s'agissait d'un document

normal. En revanche, l'interaction pour passer de l'un à l'autre de ces états nous semble moins évidente, en particulier parce que la création d'une pile de documents papier fait appel à la troisième dimension pour empiler des documents les uns par-dessus les autres. La face avant de la couverture de la pile peut également être levée à la verticale. La surface d'une table interactive est strictement bidimensionnelle. L'absence de troisième dimension fait partie des raisons que nous avons évoqué pour expliquer la complexité du pliage d'un même document numérique plus d'une fois. Ici, cette absence empêche l'utilisateur de savoir si un document va passer en dessous de la pile, au dessus de la pile, ou entre les deux faces du document plié qui sert de couverture à la pile.

Pour concevoir l'interaction d'activation de plier ou empiler, nous avons pris en compte des contraintes similaires à celles que nous avons mis en évidence pour la conception de la technique d'interaction de pliage. Nous pensons que ça ne doit pas être facile d'insérer un document dans une pile. L'utilisateur ne doit pas avoir l'impression qu'une pile a « mangé » son document alors qu'il était simplement en train de manipuler le document. Un retour visuel non ambigu doit indiquer clairement qu'un document est sur le point d'être ajouté à une pile. Un moyen d'annuler cet empilement doit y être associé. D'un autre côté, ce ne doit pas être trop difficile non plus d'ajouter un élément dans une pile. Le but de la création d'une telle pile est de supporter la création impromptue de groupe de documents. Contrairement à la création d'un dossier qui doit être planifiée, qui inclut l'attribution d'un nom, parfois même d'un nom unique, au dossier, etc., une pile supporte une organisation casuelle des documents. Dans le monde physique, l'utilisation d'un document quelconque pour servir de couverture à la pile montre que la création d'une pile se fait de manière impromptue et non planifiée. Notre technique de plier pour empiler doit pouvoir supporter la même spontanéité. Nous pensons que cela exclut l'utilisation de tâches intermédiaires ou articulatoires trop longue lors de la création de la pile.

Un retour visuel clair nous semble être un prérequis indispensable à l'accomplissement de ces buts. Ce retour peut offrir plusieurs informations. Ainsi, il peut mettre en évidence le document qui va être ajouté à une pile et la pile qui va recevoir un document. La forme physique des tables interactives fait que les deux sont importants. En effet, les tables interactives sont des ordinateurs multi-utilisateur. Plusieurs utilisateurs peuvent manipuler des documents en même temps. Si le retour ne se fait que sur la pile, les utilisateurs ne peuvent pas savoir si celui-ci concerne le document qu'ils sont en train de manipuler, ou celui d'un autre utilisateur de la table. Un utilisateur qui n'essaie pas d'ajouter son document dans une pile peut penser, en voyant un retour visuel sur une pile quelconque, que son document n'est pas concerné. Or un des buts du retour visuel est précisément d'informer les utilisateurs et de fournir un moyen d'annuler un empilement non voulu avant qu'il ne se produise. Le retour doit donc mettre en évidence quel document va être ajouté à une pile. Il doit également permettre d'identifier la pile qui va recevoir un document. La taille réduite des tables interactives actuelles fait que

plusieurs piles seront probablement proches les unes des autres. Il n'est pas impossible que le système identifie mal la pile dans laquelle l'utilisateur veut ajouter un document. Un retour visuel sur la pile permet à l'utilisateur de corriger ce problème avant qu'il ne se produise.

Nous proposons un moyen de communiquer ce double retour d'information en changeant des paramètres du document qui va être ajouté à une pile et de la pile qui va recevoir un document. Les documents manipulés par un utilisateur sont toujours affichés au dessus de tous les autres sur la table interactive. L'affichage des documents sur la surface de la table interactive se fait en deux dimensions, mais les documents ont une troisième coordonnée, l'ordre z , qui est une valeur discrète déterminant quels documents sont affichés au dessus de quels autres, les masquant ainsi partiellement. Cet ordre z n'est pas manipulable directement par l'utilisateur, il change automatiquement quand l'utilisateur interagit avec les documents. Le document actif d'un utilisateur a un ordre z supérieur à tous ceux des documents inactifs (si deux utilisateurs interagissent avec deux documents dans la même zone de la table, leurs deux documents actifs peuvent se masquer l'un l'autre ; dans ce cas, c'est le document qui a été déplacé en dernier qui est au dessus). Quand un document va être ajouté à une pile, nous modifions son ordre z pour l'afficher entre les deux faces de la pile. L'utilisateur voit que son document actif est de plus en plus masqué par un document inactif, ce qui lui donne l'information qu'il va être empilé. En revanche ce type de retour n'est pas suffisant pour indiquer à quelle pile le document va être ajouté. Plusieurs piles peuvent être proches les unes des autres sur la table interactive. Des piles peuvent aussi être superposées. Toutes les piles ayant un ordre z plus haut que la pile à laquelle le document va être ajouté seront affichées au dessus du document. Pour offrir un retour visuel sans ambiguïté, nous modifions également une propriété de la pile elle-même. La partie repliée de la couverture de la pile, comme nous l'avons proposé dans la section précédente sur le pliage, est semi transparente. Cela permet de voir les documents qui sont dans la pile pour augmenter sa reconnaissabilité. Quand un document est en train d'être glissé entre les deux faces de la couverture, nous enlevons cette transparence pour rendre la couverture complètement opaque.

Ce retour visuel, activé dès qu'un document est déplacé vers l'ouverture d'une pile, nous permet de mettre en œuvre un mécanisme pour annuler l'empilement d'un document avant qu'il ne se produise. En effet, l'empilement se fait en déplaçant les documents. Le déplacement est une interaction fondamentale. Les utilisateurs déplacent plus souvent un document pour le placer à un autre endroit de la table interactive que pour l'ajouter à une pile. Notre interaction d'empilement doit donc interférer le moins possible avec les autres types de déplacements. Nous pensons que l'annulation doit être intégrée de manière fluide avec le reste de l'interaction, en utilisant le déplacement en cours. L'utilisateur peut bien entendu continuer à déplacer le document pour l'éloigner de la pile. Mais l'utilisateur peut vouloir placer le document sur la surface de la table, à côté de la pile. Cette solution introduit des zones sur la table qui sont inaccessibles aux déplacements.

Nous proposons de casser la métaphore des côtés de la couverture. Avec une feuille de papier pliée en deux, les deux côtés sont reliés entre eux au niveau de la pliure. Une autre feuille de papier ne peut pas passer à travers cette pliure et ressortir à l'extérieur de l'autre côté. Avec des documents numériques sur une table interactive, nous ne sommes pas limités par cette réalité physique. L'utilisateur peut continuer de déplacer le document, de sorte qu'il franchisse l'arrière de la pile. Un document qui a est ressorti d'une pile ne peut plus être ajouté à cette pile pendant tout le temps du déplacement courant, ce qui permet de le déplacer très près de la pile, voire sur la pile.

3.5.2. Aspects graphiques de la métaphore

Dans la continuité de cette extension des lois physiques qui s'appliquent au papier, les piles digitales ne sont pas limitées en taille. Tout comme les vieilles librairies de Terry Pratchett [91] sont plus grandes à l'intérieur qu'à l'extérieur (différence qu'il attribue à une déformation de l'espace-temps autour des livres), nos dossiers virtuels sont infiniment profonds. Rien ne s'oppose à ce qu'une photo d'identité pliée en deux contienne un tableau de maître de dix mètres par quinze. La partie pliée de la couverture de la pile n'est pas, comme avec une feuille de papier, un obstacle physique. Nous proposons plusieurs stratégies pour afficher visuellement une telle pile.

Les documents contenus dans la pile peuvent être masqués, ou partiellement masqués, quand ils sortent de la zone de la pile. Sur la figure 46, le document le plus en haut est partiellement masqué au niveau de la pliure. Ce mécanisme de découpage (ou *clipping*) permet d'éviter qu'un document ne ressorte, visuellement, de la pile par l'arrière de la couverture. Le résultat est semblable à une pile du monde physique si les documents présents dans la pile sont de tailles cohérentes avec ce que l'on pourrait mettre dans une pile de cette taille. En revanche, si les documents ne sont pas de tailles similaires, ce rendu graphique est non réaliste (figure 49). Sur la figure 49 à gauche, le document trop profond pour la pile est stocké de façon, sinon naturelle, du moins cohérente dans cet espace infiniment grand qu'est l'intérieur de la pile. A droite, en revanche, le document est plus haut que la couverture de la pile. La même technique de *clipping* est utilisée, mais le résultat « se voit » sur le document : la frontière dépasse le cadre de la pile. La découpe nette et droite du document peut être interprétée comme un pliage du document inséré (un pliage vers l'arrière, contrairement à notre interaction de pliage vers l'avant).

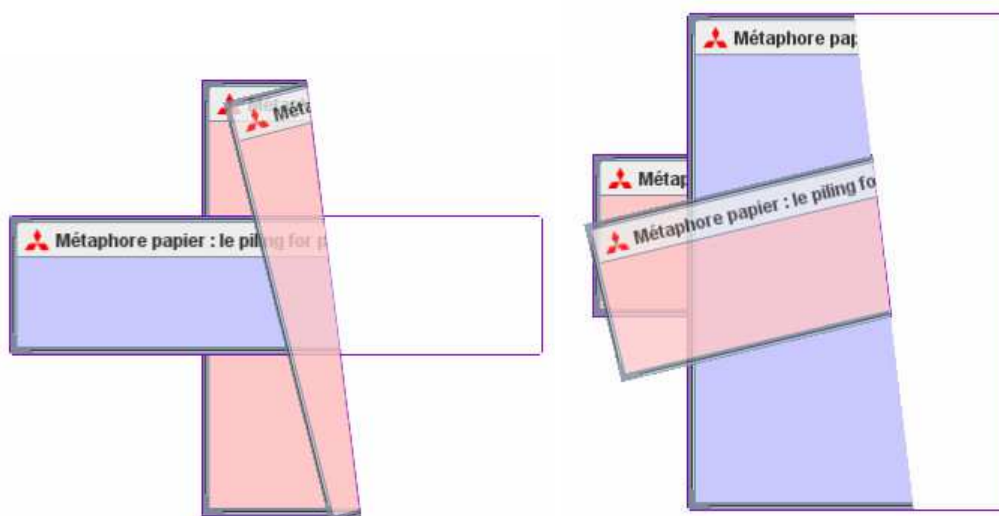


Figure 49 : Un grand document (bleu) dans une petite pile (couverture en rose). A gauche : le document est trop large ; à droite : il est trop haut. On affiche dans l'ordre le recto du document-couverture de la pile, les documents et le verso de la couverture, en découpant l'ensemble selon la droite qui passe par la pliure de la couverture. Le trait mauve indique la taille réelle du document.

Les documents insérés dans la pile peuvent également être automatiquement réduits s'ils sont trop gros. Cette réduction peut se faire sous la forme d'un redimensionnement du document, d'un changement d'échelle (zoom), ou d'une simplification sémantique similaire à [52]. Elle peut concerner l'ensemble des documents de la pile si la cohérence entre les représentations graphiques des divers documents est importante, ou concerner seulement les documents qui ne tiennent pas dans la pile. Ainsi, les documents peuvent commencer à être réduits quand ils touchent le fond de la pile. Ceci permet de créer des piles avec des documents affichés à l'échelle normale quand ils sont près de l'extérieur de la pile, et des documents de plus en plus miniaturisés à mesure qu'ils sont déplacés vers le fond de la pile. Le fond de la pile peut ainsi contenir plus de documents que la zone plus accessible du bord de la pile. La couverture et l'ensemble de la pile peuvent également être modifiés. La pile peut s'agrandir ou changer d'échelle automatiquement quand plus de place est nécessaire pour l'ajout d'un document dans la pile.

3.5.3. Implémentation

Nous avons implémenté cette technique d'interaction dans la librairie pour tables interactives DiamondSpin. La portée générale de notre technique d'interaction – n'importe quel document peut être plié et servir de couverture à une pile, sans que les développeurs d'une application donnée aient à le programmer – requiert l'implémentation d'objets spécifiques, externes aux documents. Ces objets permettent également de gérer des piles composées de documents provenant de plusieurs applications distinctes.

Nous avons nommé l'objet responsable de l'activation du plier pour empiler le DSPiler. C'est un écouteur (*listener*) qui réagit aux mouvements des documents sur la table. Il attache automatiquement une nouvelle instance de lui-même à chaque document qui récupère le focus d'un utilisateur. Plusieurs instances peuvent être actives en parallèle, car les tables interactives étant des dispositifs multi-utilisateur, plusieurs documents peuvent être au premier plan en même temps. Quand un document est déplacé, le DSPiler associé vérifie s'il y a une zone d'intersection entre le document et chaque couverture de pile potentielle. Les documents ne sont pas nécessairement carrés (par exemple, un document plié), et ne sont pas forcément alignés avec le système de coordonnées cartésien de la surface (par exemple, un document tourné ou incliné). Ceci donne lieu à des calculs d'intersection complexes, et dégradent les performances du système interactif. Nous avons trouvé que nous pouvons utiliser une approximation de ces calculs, tout en maintenant une expérience utilisateur riche, en remplaçant le test d'intersection de deux polygones arbitraires par deux tests d'intersection d'un polygone arbitraire avec un rectangle aligné avec les axes du système de coordonnées (figure 50). Nous considérons que le test d'interaction est positif si les boîtes englobantes, alignées avec les axes, de chacun des documents intersectent la forme polygonale de l'autre document. Dans la figure 50, en haut, les boîtes englobantes des deux documents s'intersectent. De plus, la boîte englobante du document plié intersecte la forme polygonale du document actif. En revanche, la boîte englobante du document actif n'intersecte pas la forme polygonale du document plié. Globalement, le test d'intersection est donc négatif. En bas, le test d'intersection est positif. Un calcul direct de la zone d'intersection aurait donné comme résultat la zone en beige. Mais ce calcul est trop complexe pour être exécuté interactivement.

Ce test d'intersection, qui est exécuté pour chaque couple document en mouvement-document plié, permet de déclencher des transitions dans une machine à états (figure 51). Pour permettre aux documents de n'entrer dans une pile que par la zone où elle est ouverte, et non par la pliure de la couverture de la pile, nous testons l'intersection du document avec chaque côté de la couverture indépendamment. Un document doit d'abord intersecter un des côtés, puis doit attendre un certain temps avant d'intersecter l'autre. Si ce délai n'est pas respecté, le document passe au dessus de la pile sans être ajouté à celle-ci. Ce temps d'attente a été paramétré empiriquement pour permettre aux utilisateurs de déplacer des documents librement même si la surface de la table interactive contient beaucoup de piles. A la vitesse à laquelle les utilisateurs déplacent usuellement les documents, le délai est trop court pour permettre l'empilement. Les utilisateurs doivent ralentir leurs mouvements pour ajouter un document dans une pile.

Le test de l'intersection du document avec chaque côté de la couverture indépendamment permet d'empêcher l'empilement d'un document dans une pile quand celui-ci s'approche de la pile par la pliure de la couverture. En effet, sur la pliure, les deux côtés de la couverture partagent une arête : ils sont donc intersectés au même moment. Mais cela permet également de « fermer » une pile la couverture

en deux parties exactement égales et superposées. Ainsi, toutes les arêtes des deux côtés sont en commun, les deux côtés sont donc toujours intersectés au même moment, et l'empilement de documents supplémentaires est interdit.

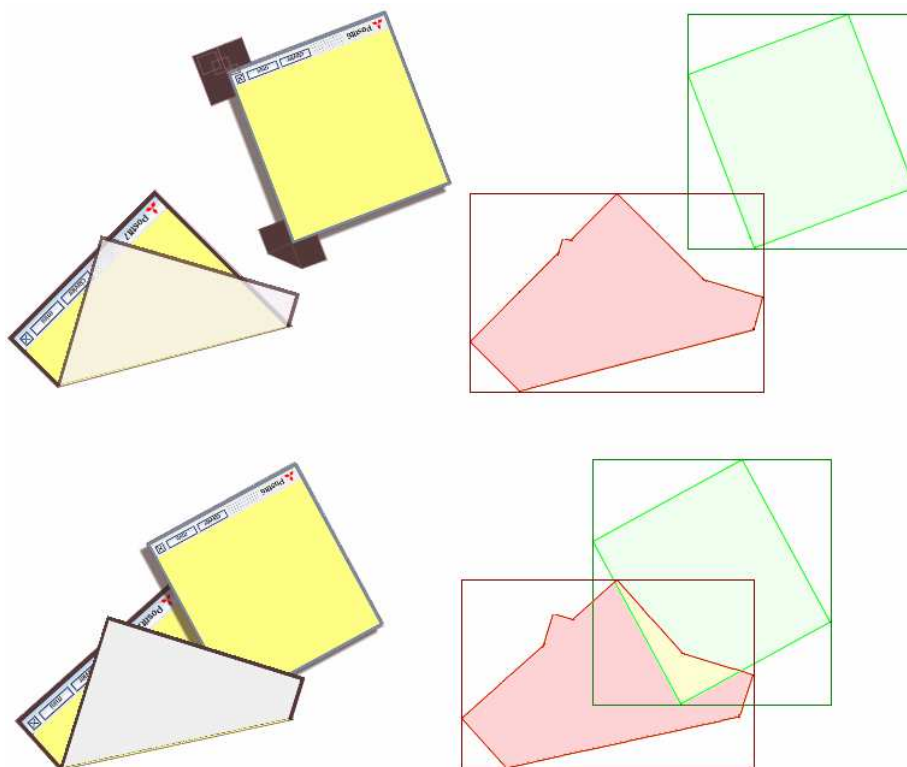


Figure 50 : Calcul de l'intersection d'un document actif (celui qui a une ombre) avec une couverture de pile potentielle. À gauche : les documents tels que vu par l'utilisateur ; à droite : une représentation symbolique de leurs formes détaillées (en vert clair pour le document manipulé et en rose pour la couverture) et de leurs boîtes englobantes alignées avec les axes du système de coordonnées de la table (en vert et rouge foncé). En haut, le document n'est pas dans la pile, et, bien que les deux boîtes englobantes s'intersectent, il est correctement calculé comme n'y étant pas ; en bas, il est dans la pile. La zone d'intersection réelle, complexe à calculer directement, est en beige.

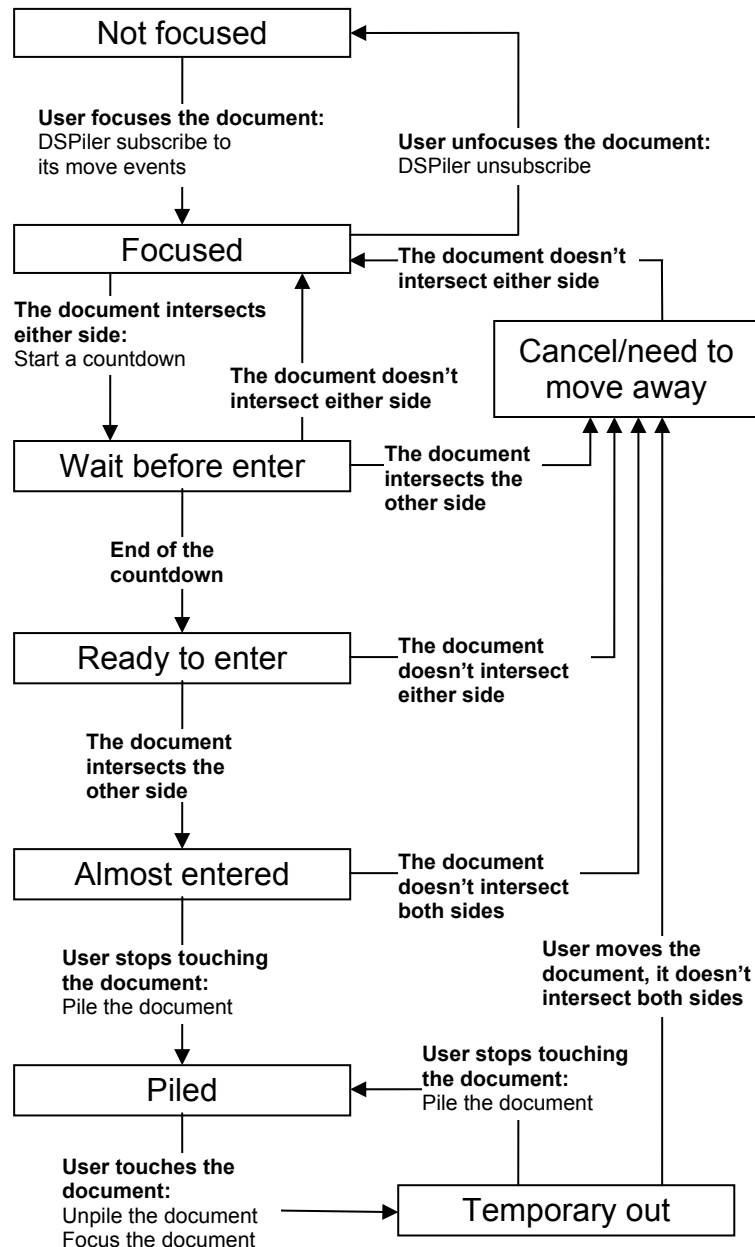


Figure 51 : Machine à états du plier pour empiler. Les côtés (*sides*) font référence à la couverture de la pile.

D'autres techniques d'interaction sont possibles pour ajouter un document à une pile, par exemple en utilisant une interaction multi-touche. L'utilisateur pourrait déplacer le document vers la pile, et simultanément « ouvrir » la pile dans laquelle il veut l'insérer en la touchant avec son autre main. Ainsi, il n'y aurait plus ni d'activation accidentelle, ni d'ambiguïté sur la pile concernée. Ceci requiert une table interactive dont le matériel informatique peut identifier l'utilisateur associé à chaque point de contact et détecter plusieurs points de contact pour un utilisateur. Pour l'instant, un tel matériel n'existe pas. La DiamondTouch peut identifier les utilisateurs, mais est limitée, dans la plupart des situations, à un point de contact par utilisateur. Entre autre, elle ne peut pas détecter plusieurs points de contact par utilisateur si les points

de contact sont trop proches géographiquement ou temporellement, ce qui serait le cas avec cette interaction.

En résumé, le plier pour empiler (*Peeling for Piling*) est une nouvelle technique d'interaction pour les tables interactives qui permet la création de piles. Elle se base sur la métaphore du papier pour offrir une interaction naturelle et intuitive. En particulier, elle permet la création de piles de façon spontanée. Ni l'utilisateur ni le développeur d'une application pour table interactive donnée n'ont besoin de planifier la création de piles ou de mettre en œuvre des mécanismes complexes au préalable. Ces créations impromptues se font en « recyclant » un document existant comme couverture pour la pile, simplement en le pliant en deux, sans utiliser de menus ou avoir besoin de saisir des données. L'insertion de documents dans la pile est aussi simple, il suffit de littéralement les déplacer dans la pile, en utilisant l'interaction de déplacement existante. Un retour visuel approprié contrôle l'ensemble du processus.

3.6. Conclusion

Nous avons présenté une métaphore pour les tables numériques interactives : la métaphore du papier. L'interaction sur une table interactive qui ne se présentent pas sur un ordinateur de bureau : plusieurs utilisateurs peuvent travailler sur plusieurs documents simultanément, ils peuvent être disposés tout autour de la table et avoir des points de vue orientés différemment, certaines parties de la surface de la table peuvent être hors d'atteinte d'un utilisateur, etc. Nous pensons qu'il est important de trouver de nouvelles sources d'inspiration pour concevoir des techniques d'interactions riches et naturelles pour les tables interactives. Les métaphores apportent ces avantages en se basant sur les connaissances existantes du monde physique qu'ont les utilisateurs.

Nous avons conçu plusieurs techniques d'interaction se basant sur cette métaphore du papier pour gérer l'espace sur la surface de la table interactive et manipuler des groupes de documents. Les fentes nous permettent de transmettre à distance des documents à un autre utilisateur, ainsi que d'effectuer des traitements sur les documents. Le pliage durable permet de nouvelles interactions avec un document. Il rend accessible le dos du document, qui peut contenir une interface graphique pour l'utilisateur. En observant le recto de documents papier existants, nous sommes arrivés à la conclusion que certains artefacts informatiques existants auraient toute leur place au dos d'un document : le système d'aide de l'application, des outils pour gérer le cycle de vie du document, des métadonnées sur le document,... Enfin, le plier pour empiler permet la création de piles de façon spontanée, en « recyclant » un document existant comme couverture pour de pile, simplement en le pliant en deux.

Ces trois interactions permettent aux utilisateurs de gérer l'espace disponible sur la table interactive. Elles peuvent s'appliquer aux logiciels pour ordinateurs de bureau existants qui sont utilisés sur une table interactive. Ainsi, nous avons présenté l'exemple de l'utilisation du logiciel existant d'édition d'images GIMP. L'interface graphique de GIMP, conçue pour un ordinateur de bureau, pose des problèmes d'utilisabilité quand elle est transposée sur une table interactive, à cause de l'utilisation de boîtes de dialogue flottantes (comme la boîte de dialogue « imprimer » sur la figure 15 au chapitre précédent), qui ne sont pas liées au document qu'elles affectent. Dans le cadre d'un utilisateur unique sur un ordinateur de bureau, celui-ci ne travaille que sur un seul document à un moment donné, les boîtes de dialogue affichées à ce moment sont donc les boîtes de dialogue concernant ce document. Dans le cadre de plusieurs utilisateurs travaillant autour d'une table interactive sur plusieurs documents, plusieurs jeux de boîtes de dialogues sont affichés sur la table. L'utilisateur ne peut pas savoir quelle boîte affecte quel document. En s'inspirant des formulaires papier qui contiennent au verso du formulaire des indications sur la manipulation de celui-ci, nous utilisons l'interaction de pliage durable pour afficher les boîtes de dialogue au dos des documents qu'elles affectent. Le lien entre la boîte de dialogue et son document n'est plus ambigu.

Nous nous attachons en particulier à mettre en avant des fonctionnalités identiques entre les interactions inclues dans les applications existantes et non adaptées aux tables interactives, et les nouvelles interactions que nous proposons. Cette façon de faire est utilisée dans plusieurs domaines de l'interaction homme-machine post-WIMP. Par exemple, les menus circulaires offrent les mêmes fonctionnalités que les menus déroulants classiques. Nous pouvons ainsi avoir des techniques d'interactions intégrées à des applications existantes de manière automatique et transparente pour l'application. Le niveau d'intégration dépendant de la flexibilité de la technologie utilisée pour faire le lien entre l'application existante et le système de table interactive. Cette flexibilité a été caractérisée au premier chapitre de cette thèse.

CHAPITRE 4

Différents moyens d'activer
différentes interactions : la
matrice des interactions _____

4.1. Introduction

Certaines des interactions que nous avons présentées au chapitre précédant peuvent être effectuées de plusieurs manières. Par exemple, le pliage d'un document peut se faire à l'aide d'un interacteur dédié, ou d'un geste du doigt, ou avec un moteur de simulation de physique et de collisions. Nous observons la même propriété avec le papier physique, et en général avec les objets du monde physique. Dans le monde physique, les différentes façons d'interagir avec un objet sont disponibles simultanément. Ainsi, une feuille de papier peut être pliée en utilisant une ou deux mains, en pinçant un coin de la feuille entre deux doigts ou en utilisant toute la main pour saisir la feuille par le centre (figure 52). Il est intéressant d'étudier si, similairement, les différentes façons d'obtenir un résultat donné avec un artefact numérique peuvent être utilisées simultanément.

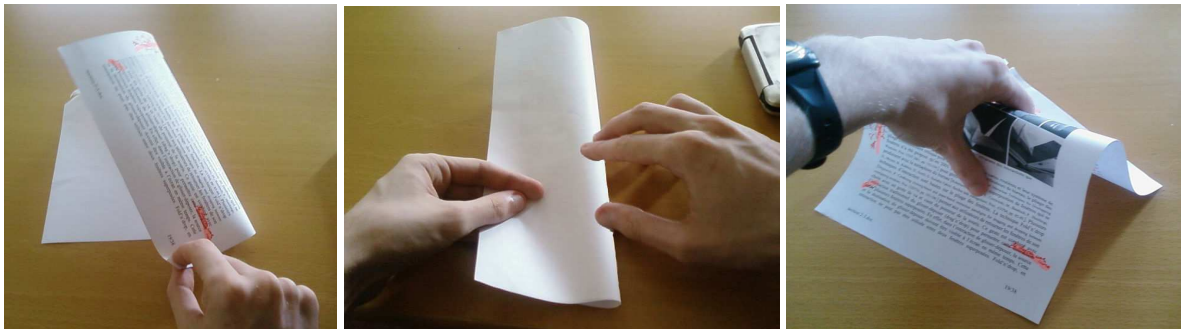


Figure 52 : Il y a plusieurs façons d'effectuer l'interaction « pliage » avec du papier physique. Les différentes techniques sont toutes disponibles simultanément.

D'autre part, différentes façons d'effectuer différentes interactions sont parfois similaires. Ainsi, avec la boîte à outils DiamondSpin, les utilisateurs disposent d'interacteurs dans les coins des documents pour redimensionner, changer leur facteur d'échelle, les tourner et les plier. Ces quatre interactions peuvent également être effectuées d'autre façons : le redimensionnement peut utiliser un geste à deux doigts, la rotation peut utiliser la trajectoire du doigt, etc. Nous avons précédemment proposé la métaphore du papier pour augmenter l'utilisabilité des tables interactives, en permettant aux utilisateurs d'appliquer leurs connaissances du monde physique aux interactions avec les artefacts numériques. Nous pensons qu'il est également important que les utilisateurs puissent transférer latéralement les connaissances d'une technique d'interaction pour table interactive pour découvrir une autre technique d'interaction. Par exemple, une fois que l'utilisateur a compris le système de redimensionnement d'un document en touchant et en déplaçant un coin spécifique, il peut découvrir les interactions de changement d'échelle, de rotation et de pliage en essayant de toucher et de déplacer les autres coins du document.

Certaines façons d'utiliser des interactions peuvent être incompatibles avec certains types de documents. Nous avons déjà évoqué que nous travaillons avec des documents qui peuvent être issus d'applications existantes, et qui peuvent contenir

des éléments d'interface utilisateur graphique. Ceci n'est pas compatible avec les interactions qui proposent d'utiliser l'intérieur des documents pour déclencher les interactions. Avec des fenêtres qui contiennent une interface utilisateur, si l'utilisateur clique à l'intérieur de la fenêtre, il va activer un composant de cette interface. De nombreux travaux s'intéressent ainsi à la manipulation de photographies ou de documents statiques, qui ne pose pas ce problème. Le type de document influe donc sur les possibilités d'activer les techniques d'interaction. Certaines façons d'utiliser des interactions peuvent également être incompatibles entre elles. Par exemple un geste ou l'espacement entre deux doigts contrôle le changement de l'échelle d'un document ne peut pas également être utilisé pour changer sa taille. En revanche, le barycentre de deux doigts peut être utilisé pour changer simultanément la position du document.

Pour faire apparaître ces contraintes, nous proposons de présenter diverses techniques d'interaction et diverses façons de les utiliser sous la forme d'une matrice mettant en correspondance des actions ou des résultats avec des moyens d'effectuer ces actions ou d'obtenir ces résultats.

Nous nous concentrons sur les interactions déjà existantes dans la gestion des fenêtres sur les ordinateurs de bureau : le déplacement et le redimensionnement, auxquelles nous ajoutons trois interactions : la rotation, le changement d'échelle et le pliage. Ces interactions s'approchent de devenir des standards en matière d'interface pour table interactive dans la communauté des tablettops [104, 93, 10, 107]. L'orientation a été identifiée comme un point sensible de l'interaction avec les tables interactives dès 1988 [61]. L'orientation et le pliage ne sont pas non plus des interactions spécifiques aux tables interactives. Elles ont également été proposées dans les interfaces utilisateurs d'ordinateurs de bureau [6, 98] pour permettre d'améliorer les interactions avec des fenêtres superposées. Le changement d'échelle, quand à lui, est utilisé depuis longtemps pour permettre entre autre une meilleure accessibilité des ordinateurs aux personnes visuellement déficientes. Il est intégré dans les systèmes d'exploitation grand public, pour changer l'échelle aussi bien de l'ensemble de l'affichage que d'une sous partie (par exemple changer la taille des icônes d'un dossier). Les différentes actions forment les colonnes de la matrice (figure 53).

Action Moyen	Déplacement	Rotation	Redimen- sionnement	Changement d'échelle	Pliage
-----------------	-------------	----------	------------------------	-------------------------	--------

Figure 53 : Les différentes actions que nous considérons.

4.2. Interacteur dédié

Le premier moyen d'effectuer toutes ces actions a été introduit dans le chapitre précédent. Il s'agit d'utiliser un interacteur dédié placé à l'extérieur de la fenêtre. Cette méthode est compatible avec les documents d'applications existantes, car elle n'interfère pas avec le contenu des fenêtres. Nous avons choisi de placer les interacteurs près des coins des fenêtres. Les coins de la bordure des fenêtres sont une zone privilégiée pour le placement d'interacteurs dans les environnements d'ordinateurs de bureau (figure 32 au chapitre précédent). Les coins offrent également une plus grande surface, ce qui permet une interaction digitale (i.e. au doigt) plus facile. L'aspect visuel des interacteurs correspond à leur fonction. Ils fonctionnent tous de la même façon : l'utilisateur touche l'interacteur avec son doigt, puis déplace son doigt. L'interacteur suit alors le doigt, modifiant la fenêtre en conséquence. L'aspect visuel associé à cette uniformité permet à l'utilisateur qui sait manipuler un des coins de deviner le fonctionnement des autres.

Le coin de redimensionnement est placé en bas à droite des fenêtres, à l'emplacement de la poignée de redimensionnement dans de nombreux environnements graphiques pour ordinateur de bureau (zone a sur la figure 32). Nous pensons que les utilisateurs peuvent utiliser leurs connaissances du redimensionnement des fenêtres sur les ordinateurs de bureau pour découvrir le coin de redimensionnement sur les tables interactives, puis l'ensemble des coins d'interaction pour la rotation, le changement d'échelle et le pliage.

Enfin, le déplacement des fenêtres utilise un interacteur dédié, mais d'une forme différente : la barre de titre de la fenêtre. En déplaçant, avec son doigt, la barre de titre sur la surface de la table, l'utilisateur déplace également l'ensemble de la fenêtre. Concernant les applications existantes, toutes les technologies présentées au premier chapitre permettent d'obtenir le contenu des fenêtres d'une application sans les décorations (barre de titre, bordures, menus,...) ou de séparer ces décorations du contenu. Nous pouvons ainsi substituer des barres de titres adaptées aux tables interactives. Nous proposons d'utiliser deux barres de titres pour chaque fenêtre : une en haut de la fenêtre et une en bas de la fenêtre. L'utilisation de l'interaction directement avec le doigt pose des problèmes d'occlusion du contenu affichée sur la table interactive. Quand l'utilisateur manipule la barre de titre en haut de la fenêtre, sa main masque le contenu de la fenêtre (figure 54, en haut à gauche). La barre de titre placée en bas de la fenêtre permet de déplacer la fenêtre sans la masquer avec sa main (figure 54, à droite). Ceci s'applique au déplacement de fenêtres qui sont orientées dans le sens de lecture de l'utilisateur qui les manipulent. L'utilisateur peut également vouloir manipuler des fenêtres qui ne sont pas orientées dans son sens de lecture, par exemple pour montrer une fenêtre à l'utilisateur qui est en face de lui, ou saisir une des fenêtres de ce même utilisateur. Dans ce cas, c'est l'utilisation exclusive d'une barre de titre en bas de la fenêtre qui provoquera la même situation d'occlusion (figure 54, en bas à gauche). Nous utilisons donc deux barres de titres simultanément : l'une pour manipuler ses propres fenêtres pour soi-même, l'autre

pour manipuler ses fenêtres en interaction avec les autres utilisateurs ou pour manipuler leurs fenêtres.

L'utilisation d'un interacteur dédié, le premier moyen que nous avons identifié pour effectuer nos cinq actions de gestion de fenêtres sur une table interactive, est résumée dans la matrice de la figure 55.

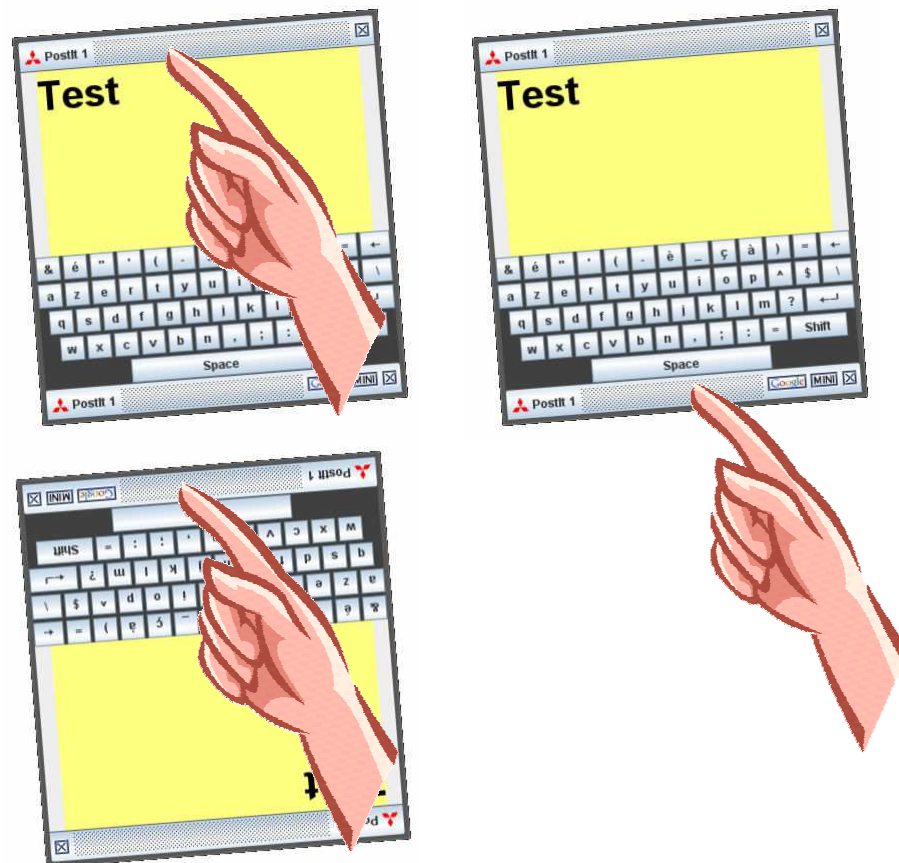


Figure 54 : Deux barres de titres pour éviter l'occlusion du contenu des fenêtres par la main de l'utilisateur lors de diverses situations de manipulation des fenêtres.

Action	Déplacement	Rotation	Redimensionnement	Changement d'échelle	Pliage
Moyen					
Interacteur dédié	et/ou				

Figure 55 : Matrice de l'utilisation d'un interacteur dédié.

4.3. Position du document sur la table

Le second moyen que nous considérons est d'utiliser la position du document lors de son déplacement pour modifier d'autres caractéristiques du document. Dans le chapitre précédent, nous avons proposé de plier un document lorsqu'il entrait en collision avec le bord de la table. Le déplacement du document vers le bord de la table permet donc d'activer l'interaction de pliage. D'autres actions peuvent être effectuées en déplaçant le document à modifier. DiamondSpin inclut un mode qui change le facteur d'échelle des documents en fonction de leur distance au centre de la table. La distance d'une zone de la table au centre de la table influe de façon significative sur l'usage qui est fait de cette zone [103]. Les bords de la table sont utilisés pour les activités individuelles, tandis que le centre est utilisé comme zone de stockage ou pour les activités collaboratives, selon l'application. Pour les applications plus individuelles, les documents proches du centre de la table sont affichés de plus en plus petits. Le centre de la table, utilisé comme lieu de stockage et de partage de document par les utilisateurs travaillant indépendamment, peut ainsi contenir plus de documents visibles à la fois. Ce mode s'appelle le mode « *black-hole* ». Pour les applications collaboratives, en revanche, le centre de la table sert de support au travail commun. Nous utilisons dans ce cas là le mode « *fish-eye* », dans lequel les

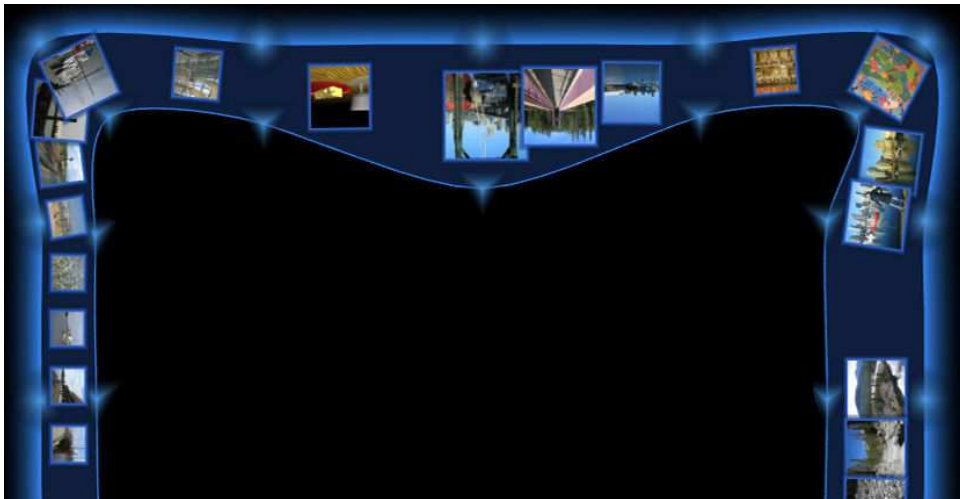


Figure 56 : L'environnement pour tables interactives *Interface Currents* change l'échelle des documents en fonction de leurs positions [44].

documents proches du centre de la table sont affichés de plus en plus grands. Cela permet une meilleure lisibilité et une manipulation au doigt plus facile dans cette zone physiquement éloignée des utilisateurs.

Un concept similaire est utilisé dans l'environnement pour tables interactives *Interface Currents* [44] (figure 56). Les documents circulent dans un bandeau plus ou moins large, et change d'échelle en fonction de la largeur du bandeau. Les documents peuvent circuler automatiquement le long du bandeau, ou être déplacés directement par les utilisateurs. Dans l'exemple de la figure 56, deux zones plus larges du bandeau sont placées en face de deux utilisateurs (en haut et en bas), tandis que le bandeau est plus fin sur les deux autres cotés de la table (spécialement à gauche). Les documents sont donc affichés en plus grands devant les utilisateurs.

La position d'un document sur la table peut également être utilisée pour changer l'orientation du document. L'orientation des documents est un problème important sur les tables interactives [61], et nous pensons qu'il est utile de fournir un mécanisme pour orienter automatiquement les documents dans le sens qui a le plus de chance d'être celui que l'utilisateur désire. Par exemple, le document pourrait tourner automatiquement son bas (au sens de « bas de page ») vers l'utilisateur quand celui-ci l'édite, et tourner son haut vers le centre de la table quand il est montré à quelqu'un d'autre ou passé à un autre utilisateur. La rotation automatique vers le centre de la table est implémentée dans DiamondSpin à l'aide d'un système de coordonnées polaires [41].

Pour déterminer la position de l'utilisateur, nous utilisons les barres d'outils de DiamondSpin. Chaque utilisateur possède une barre d'outils qu'il peut faire glisser le long du bord de la table. Initialement, les barres d'outils sont placées autour de la table, en face des chaises. Nous faisons l'hypothèse que si un utilisateur change de place, de manière plus ou moins définitive, il va « emporter » sa barre d'outils jusqu'à sa nouvelle position. Elle contient plusieurs éléments essentiels de l'interface dont il a besoin pour travailler. La barre d'outils nous donne une bonne approximation de la direction de l'utilisateur autour de la table, mais ne nous donne pas le point vers lequel orienter les documents. Nos résultats expérimentaux suggèrent de placer ce point en dehors de la table, de manière à ce que les documents confinés dans la surface de la table, s'orientent d'environ $\pm 45^\circ$ autour de l'axe utilisateur-centre de la table. Nous trouvons que la position qui semble la plus agréable par rapport à la façon dont les utilisateurs se placent autour de la table est obtenue en prenant le centre de la barre d'outils et en la déplaçant de la moitié du rayon du cercle inscrit dans la surface de la table. Nous effectuons ce décalage, non pas dans l'alignement centre de la table – centre de la barre d'outils, mais perpendiculairement au bord sur lequel est située la barre d'outils (figure 57). En effet, les utilisateurs se placent autour de la table interactive comme autour d'une table carrée (assis parallèles au bord), et non comme autour d'une table ronde (assis en regardant au centre). Les travaux de Wigdor *et al.* nous apprennent de plus que les utilisateurs choisissent la position la plus confortable et non la plus efficace pour la tâche [126]. Il nous semble donc important de nous adapter à cette façon de s'asseoir.

travail (i.e. construire et organiser l'arbre) et laisser à la machine le soin d'aligner et d'orienter les documents. Certaines opérations de grande ampleur, nécessitant par exemple le déplacement de toute une sous-branche de l'arbre, s'en trouvent simplifiées. Elles ont donc plus de chance d'être entreprises, quitte à ensuite faire marche arrière si l'opération ne contribuait finalement pas à la tâche de construction de l'arbre.

D'autre part l'interaction permet de désambiguïser certaines relations inter-documents : lorsqu'un document est aligné avec deux autres (comme l'ovale bleu tout à fait à droite la figure 58, qui aligné avec les deux carrés verts à mi-hauteur de la figure selon les axes rouges et bleus), le déplacer légèrement (par exemple le long de la flèche noire) permet de constater vers lequel il se réoriente (le deuxième axe bleu). Ces petites interactions d'obtention d'informations permettent de rendre plus naturelles les interactions de modifications de la structure, par exemple, pour un utilisateur qui arriverait en cours d'une séance de construction collaborative d'un arbre sur une table interactive.

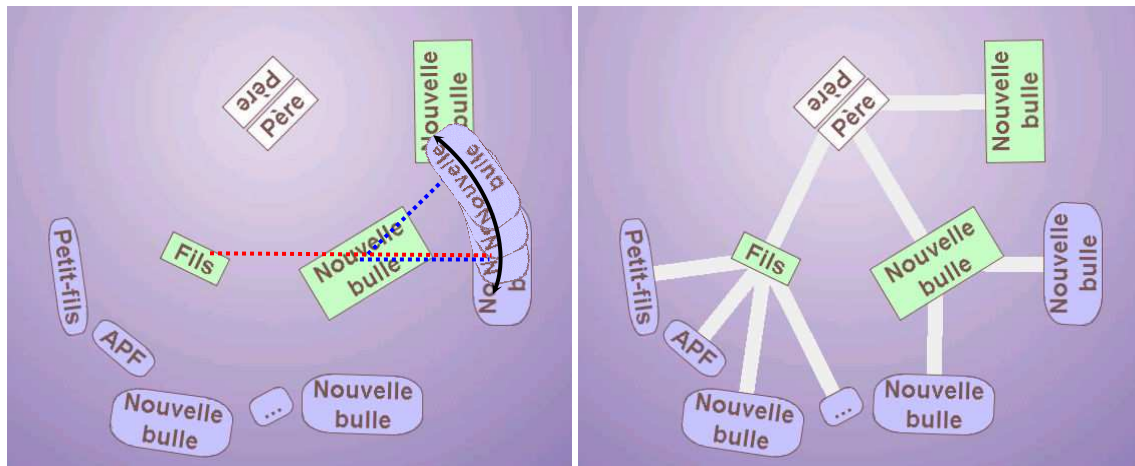


Figure 58 : Les utilisateurs doivent construire et organiser un arbre. Les nœuds orientent leur haut vers leur père, et, lorsque l'utilisateur les déplace, donnent ainsi l'impression qu'ils leur tournent autour. À droite : les liens sont visuellement représentés ; à gauche : ils sont suggérés par l'orientation et l'interaction de déplacement.

Utiliser la position du document sur la table pour le déplacer, i.e. modifier sa position, n'a pas de sens. La position du document est la position du document. La case concernant l'interaction de déplacement est donc vide pour ce moyen. En revanche, nous pouvons le rapprocher de notre travail sur les fentes de transmission. Dans le cas des fentes, ce n'est pas directement la position du document qui permet de le déplacer à un autre endroit, mais sa présente dans la zone d'effet de la fente.

Enfin, la position d'un document pourrait être utilisée pour changer ses dimensions, mais nous ne voyons pas d'applications où une telle méthode pour redimensionner un document serait utile. Sur les ordinateurs de bureau, cette méthode est utilisée dans Windows 7 quand l'utilisateur déplace un document tout en haut de l'écran ou à l'extrême droite ou extrême gauche de l'écran. Dans le premier cas, le document est

redimensionné pour occuper toute la surface de l'écran, dans les deux autres cas, seule la dimension verticale du document est modifiée pour que celui-ci prenne toute la hauteur disponible tout en gardant la même largeur.

L'utilisation de la position du document pour modifier d'autres caractéristiques de celui-ci, le second moyen que nous avons identifié pour effectuer des actions de gestion de fenêtres sur une table interactive, est résumée dans la matrice de la figure 59. L'utilisation de la position du document est compatible avec l'utilisation simultanée des coins d'interaction. Ainsi, nous pouvons proposer un état par défaut du document (une orientation par défaut, un zoom par défaut,...) en fonction de sa position sur la table, qui nous informe dans une certaine mesure sur la façon dont l'utilisateur utilise ce document (modification, stockage, activité collaborative,...). Si cet état par défaut n'est pas adapté à l'activité de l'utilisateur, ce dernier peut utiliser les coins pour changer l'orientation, le zoom,... du document manuellement. L'utilisation de la position du document est également compatible avec les documents d'applications existantes, car cela n'interfère pas avec le contenu du document.

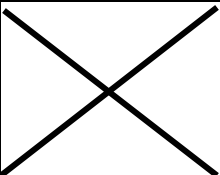
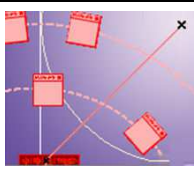
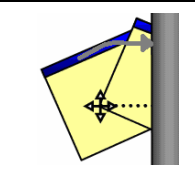
Action Moyen	Déplacement	Rotation	Redimension- nement	Changement d'échelle	Pliage
Position sur la table					

Figure 59 : Matrice de l'utilisation de la position de la fenêtre pour modifier d'autres caractéristiques de celle-ci.

4.4. Trajectoire du doigt

Enfin, le troisième moyen d'activer les interactions de gestion des documents sur la surface de la table est l'utilisation de la trajectoire⁶ du doigt. Nous avons déjà abordé, au chapitre précédant, l'utilisation d'un mouvement rapide du doigt sur la bordure d'un document pour plier celui-ci. Nous avons jugé que cette technique est un moins bon compromis entre la nécessité d'une interaction métaphorique et les besoins d'une interface utilisateur utilisable. Elle permet de plier le document plus librement (n'importe quelle forme peut être facilement obtenue), mais elle n'est pas aussi découvrable que l'utilisation d'un interacteur, visuel, dédié. Par exemple, plusieurs utilisateurs de la démonstration de *Fold'n'drop* [25], qui utilise un mouvement de la souris similaire à notre mouvement du doigt, n'ont tout simplement pas découvert la nouvelle interaction, et l'ont utilisé comme leur système habituel.

Comme nous l'avons expliqué en introduction de ce chapitre, nous pensons qu'il est possible que les utilisateurs puissent transférer les connaissances d'une technique d'interaction pour table interactive pour découvrir une autre technique d'interaction, de la même manière que la métaphore du papier se base sur le concept que les utilisateurs appliquent leurs connaissances du monde physique aux interactions avec les artefacts numérique. Ainsi, en créant d'autres interactions utilisant la trajectoire du doigt, nous pouvons augmenter la découvrabilité de ce moyen de plier les documents.

La trajectoire du doigt a été couplé à l'interaction de déplacement pour former l'interaction de *Rotate'n'Translate* (RnT) [61]. Le document est déplacé avec le doigt, mais il « frotte » sur la table, selon une formule inspirée des équations de friction en physique. Contrairement à l'utilisation de la position absolue du document sur la table que nous avons explorée précédemment, c'est ici le chemin suivi par le document qui détermine son orientation, et la trajectoire du document suit évidemment celle du doigt qui le déplace. Dans [61], Kruger *et al.* proposent de toucher le document en un emplacement quelconque pour le déplacer. Ils peuvent ainsi, en touchant le document en bas de celui-ci et en déplaçant vers soi, l'orienter dans la direction de lecture de l'utilisateur (figure 60 en bas). La même orientation peut être obtenue en touchant le haut du document et en l'éloignant de soi (figure 60 en haut), ou en touchant une zone sur la droite du document et le déplaçant vers sa droite, etc. Le document « frottant » sur la table, l'axe centre du document-point où l'utilisateur à toucher s'aligne avec la direction du déplacement.

⁶ Nous n'utilisons pas le mot « geste » car il peut référer de manière ambiguë à deux concepts différents. Il peut s'agir d'un geste statique de la main (appelé en anglais « *posture* » par Chia Shen [108b]), par exemple placer la tranche de la main sur la table pour délimiter une frontière, ou placer la paume de la main à plat sur la table pour signifier « stop ». Il peut également s'agir d'un mouvement d'un/des doigts et/ou d'une/des mains (appelé en anglais « *gesture* » par Chia Shen), par exemple écarter deux doigts pour changer l'échelle d'un document, ou frotter la table avec la paume de la main pour signifier « effacer ». La trajectoire du doigt, quand à elle, est un sous-type de « *gesture* » dans le quel seul un doigt intervient.

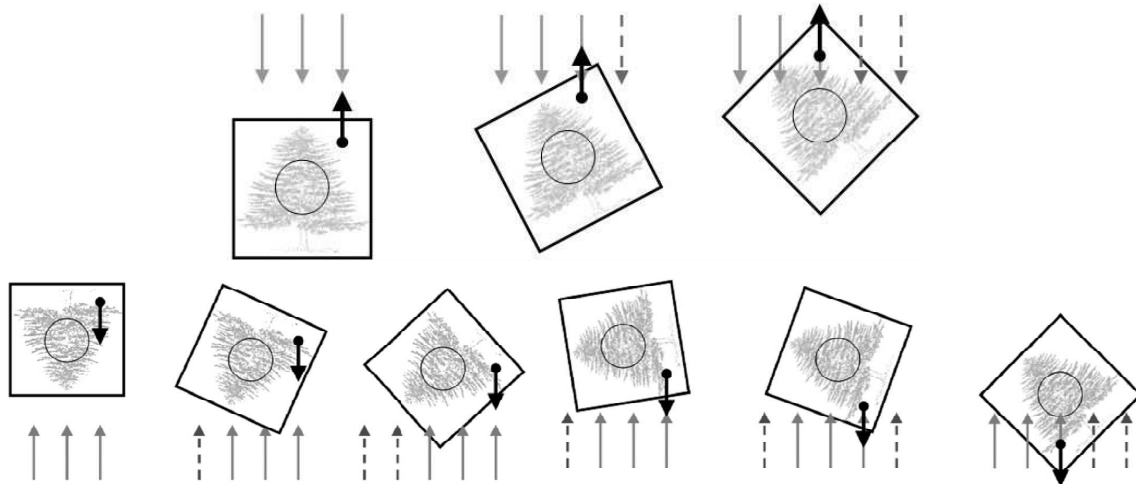


Figure 60 : L'utilisation de forces de friction (flèches en gris), combinée à une force de déplacement appliquée sur le bord du document (flèche en noir), permet de tourner le document selon la trajectoire du déplacement [61].

Une interaction très similaire appelée TNT a été proposée par Liu *et al.* [66]. Ils se basent spécifiquement sur une analogie avec le déplacement d'une feuille de papier sur une table (figure 61).

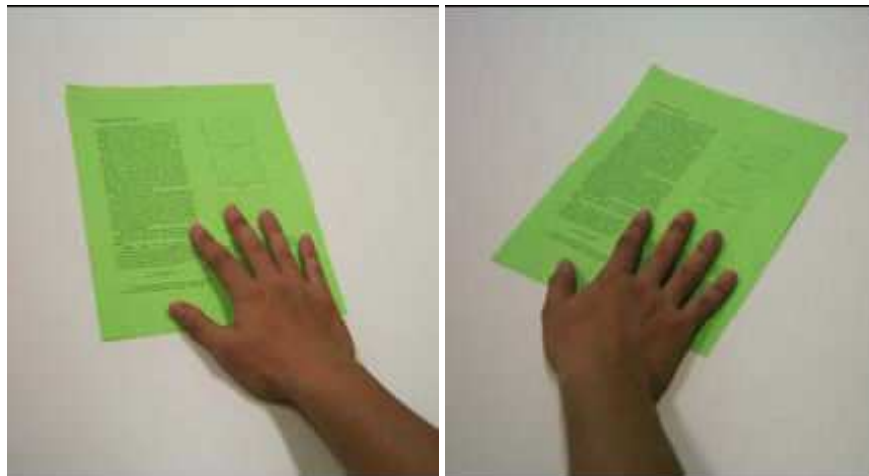


Figure 61 : L'interaction TNT, qui permet de réorienter un document en le déplaçant, se base sur une analogie avec le déplacement d'une feuille de papier sur une table [66].

Cependant, ces interactions posent problème quand le document n'est pas une image statique. En effet, l'utilisateur ne peut pas toucher l'intérieur du document pour le déplacer, car il activerait alors des composants de l'interface graphique qui constitue le contenu du document. Nous proposons une alternative à RnT et à TNT qui ne dépend pas de l'emplacement où l'utilisateur a touché le document, mais uniquement de la trajectoire du déplacement. Le système calcule la tangente à la trajectoire au niveau du doigt de l'utilisateur, et utilise la direction de cette tangente et le sens du mouvement pour orienter le document (figure 62). Pour obtenir

l'orientation finale souhaitée, l'utilisateur peut déplacer le document plus loin que nécessaire puis revenir en arrière avec l'angle approprié (figure 62, à droite). Dans nos tests informels, les utilisateurs ont adopté ce comportement spontanément.

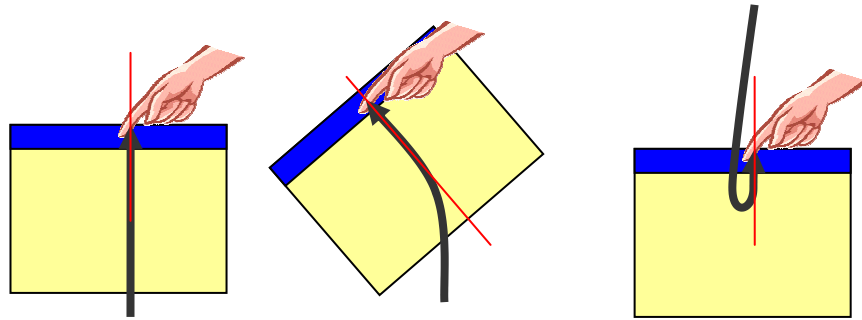


Figure 62 : Déplacement et orientation d'un document en un seul mouvement. La tangente (en rouge) à la trajectoire (en noir) au niveau du doigt de l'utilisateur est utilisée pour orienter le document.

Cette interaction est facile à découvrir, car elle s'active dès que l'utilisateur déplace un document. Elle peut permettre à l'utilisateur de chercher d'autres interactions utilisant la trajectoire du doigt.

La trajectoire du doigt peut également servir à déplacer les documents avec une interaction de jeter de documents. Nous avons évoqué précédemment que nous avons implémenté un modèle physique de l'inertie et des frottements, de telle manière à ce que quand un utilisateur arrête de toucher un document qu'il était en train de déplacer, le document continue de se déplacer avec la « force cinétique » qu'il a emmagasiné. La vitesse et la direction du mouvement du doigt de l'utilisateur au moment où celui-ci cesse de toucher le document sont utilisées pour déterminer la distance que le document va continuer à parcourir et la direction de ce déplacement, respectivement. Nous avons paramétré empiriquement la fonction entre la vitesse du doigt et la distance du lancer pour que les documents lancés par un mouvement très vif et sec du doigt parcourent l'équivalent de la diagonale de la table. Ainsi, les utilisateurs peuvent envoyer des documents d'un bout à l'autre de la table, mais ne peuvent pas faire du « ping-pong » avec les documents (les documents rebondissent sur les bords de la table pour ne pas sortir de la surface interactive). L'interaction de lancer nous semble toutefois approximative et peu pratique. Lors de nos essais, les utilisateurs ont eu des difficultés à placer le document à l'endroit désiré. Les problèmes viennent autant de la déviation angulaire (une légère déviation lors du lancer produit un large écart à grande distance), que du dosage de la force nécessaire pour parcourir la distance voulue et ni plus, ni moins.

Le changement d'échelle et le redimensionnement peuvent utiliser un geste similaire à celui que nous avons proposé pour le pliage. L'utilisateur peut effectuer un geste d'aller-retour rapide sur le bord du document, suite à quoi le bord du document suit le doigt. En revanche, ces trois interactions ne peuvent pas être utilisées en même temps sans un mécanisme supplémentaire pour choisir l'interaction à activer. Il est

aussi à noter que certains types de documents n'ont pas de bordures, et ne sont donc pas compatibles avec cette interaction. L'utilisation de la trajectoire du doigt, le troisième moyen que nous avons identifié pour effectuer des actions de gestion de fenêtres sur une table interactive, est résumée dans la matrice de la figure 63.

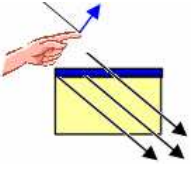
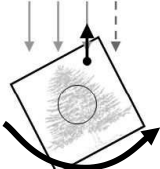
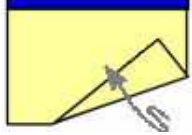
Action Moyen	Déplacement	Rotation	Redimension- nement	Changement d'échelle	Pliage
Trajec- toire du doigt			En remplacement du pliage	En remplacement du pliage	

Figure 63 : Matrice de l'utilisation de la trajectoire du doigt.

4.5. Conclusion

De nombreux autres moyens peuvent être envisagés pour effectuer ces actions. Nous pouvons citer l'utilisation de l'intégralité de la main de l'utilisateur dans *RoomPlanner* [129]. Epps *et al.* [25] ont demandé à des utilisateurs d'effectuer des actions sur une table interactive sans leur donner d'instructions sur la manière de procéder. Le corpus de gestes ainsi obtenu, contenant les gestes spontanément choisis par les utilisateurs, montre une remarquable correspondance entre certaines actions et des gestes effectués spontanément par les différents utilisateurs. Ainsi, plus de 90% de ceux-ci ont utilisé le doigt pour dessiner, et plus de 80% pour déplacer un slider. Des gestes plus complexes se retrouvent de façon identique chez plusieurs utilisateurs, qui n'ont pas d'a priori sur la façon dont fonctionne le système. Par exemple, le geste « attraper en fermant la main, puis relâcher en ouvrant la main » pour l'action de copier-coller.

L'interaction bi-manuelle est intéressante, mais malheureusement relativement limitée voire difficilement possible avec les tables DiamondTouch que nous avons utilisé. L'utilisation des deux mains permet d'augmenter les performances des utilisateurs [14]. Buxton *et al.* notent que les utilisateurs novices ont spontanément adopté des stratégies de parallélisation pour effectuer deux sous-tâches en même temps. L'augmentation des performances concerne aussi les tâches séquentielles, mais dans une moindre mesure. La *Magic Table* de François Bérard [8] utilise une interaction avec deux points de contrôle indépendants pour simultanément déplacer, orienter et changer d'échelle d'un document (figure 64).

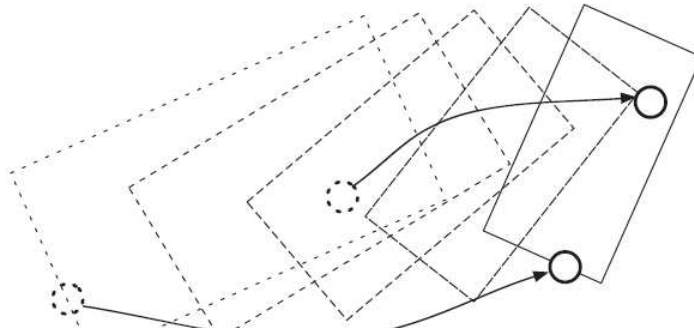


Figure 64 : Une interaction bi-manuelle utilisant deux points de contrôle indépendants pour manipuler simultanément la position, l'échelle et l'orientation d'un document [8].

Nous résumons dans la figure 65 ci-dessous les trois moyens que nous avons présentés pour effectuer cinq actions de gestion de fenêtres sur une table interactive : le déplacement, la rotation, le redimensionnement, le changement d'échelle et le pliage. Ils sont présentés sous la forme d'une matrice faisant correspondre pour chaque action et chaque moyen une technique d'interaction. Dans le monde physique, et particulièrement avec le papier dont nous avons développé la métaphore

tout au long de cette thèse, nous observons qu'une action peut être effectuée de plusieurs façons différentes. De plus, les différentes façons d'interagir avec un objet sont disponibles simultanément. Notre matrice met en évidence que cette propriété peut être étendue aux artefacts numériques. Les différentes façons de plier un document numérique ou de le réorienter peuvent être utilisées en parallèle. Par exemple, une technique passive comme l'utilisation de la position sur la table, peut fournir une valeur par défaut à l'orientation du document, tandis qu'une technique active, comme l'utilisation d'interacteurs dédiés, peut permettre aux utilisateurs de changer cette valeur par défaut.

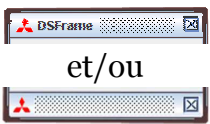
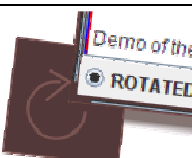
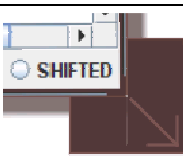
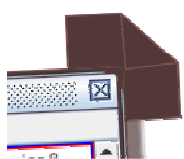
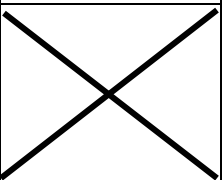
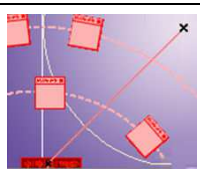
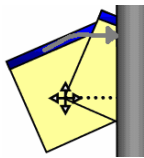
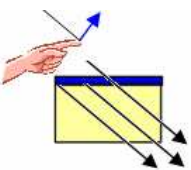
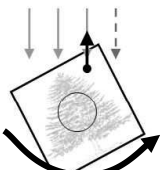
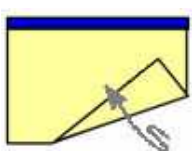
Action Moyen	Déplacement	Rotation	Redimensionnement	Changement d'échelle	Pliage
Interacteur dédié					
Position sur la table					
Trajectoire du doigt			En remplacement du pliage	En remplacement du pliage	

Figure 65 : Différents moyens d'activer différentes interactions : la matrice des interactions que nous avons présentée dans ce chapitre et dans cette thèse.

L'utilisation d'une métaphore pour concevoir des interactions homme-machine permet aux utilisateurs d'appliquer leurs connaissances du monde physique aux interactions avec les artefacts numériques. Les interactions sont plus facilement découvrables et compréhensibles. De la même manière, nous pensons que d'utiliser des moyens similaires pour effectuer plusieurs actions doit permettre aux utilisateurs de transférer latéralement les connaissances d'une technique d'interaction pour table interactive pour découvrir une autre technique d'interaction. Par exemple, une fois que l'utilisateur a compris le système de redimensionnement d'un document en touchant et en déplaçant un coin spécifique, il peut découvrir les interactions de changement d'échelle, de rotation et de pliage en essayant de toucher et de déplacer les autres coins du document.

Enfin, une case de la matrice n'est pas complétée : le redimensionnement en utilisant la position sur la table. D'autres cases pourraient ne pas être complétées quand nous

étudierons plus de moyens et plus d'actions. Ces cases peuvent inspirer le concepteur de techniques d'interaction pour générer de nouvelles techniques en combinant des actions et des moyens existants. Ainsi, dans le cas du redimensionnement en utilisant la position sur la table, une technique similaire existe dans l'environnement pour ordinateurs de bureau de Windows 7 lorsque l'utilisateur déplace un document tout en haut de l'écran ou à l'extrême droite ou extrême gauche de l'écran. Nous pourrions ajouter une troisième dimension à la matrice (figure 66) pour les dispositifs d'interaction : ordinateurs de bureau, tables interactives, périphériques portables,... A mesure que les dispositifs auxquels a accès l'utilisateur se démultiplient, le chercheur en interaction homme-machine peut vouloir comparer comment des actions et des moyens sont mis en œuvre sur les divers dispositifs interactifs. A nouveau, il est logique de penser que l'utilisateur d'un nouveau type de dispositifs peut se baser sur ses connaissances du fonctionnement de dispositifs similaires pour découvrir les techniques d'interaction.

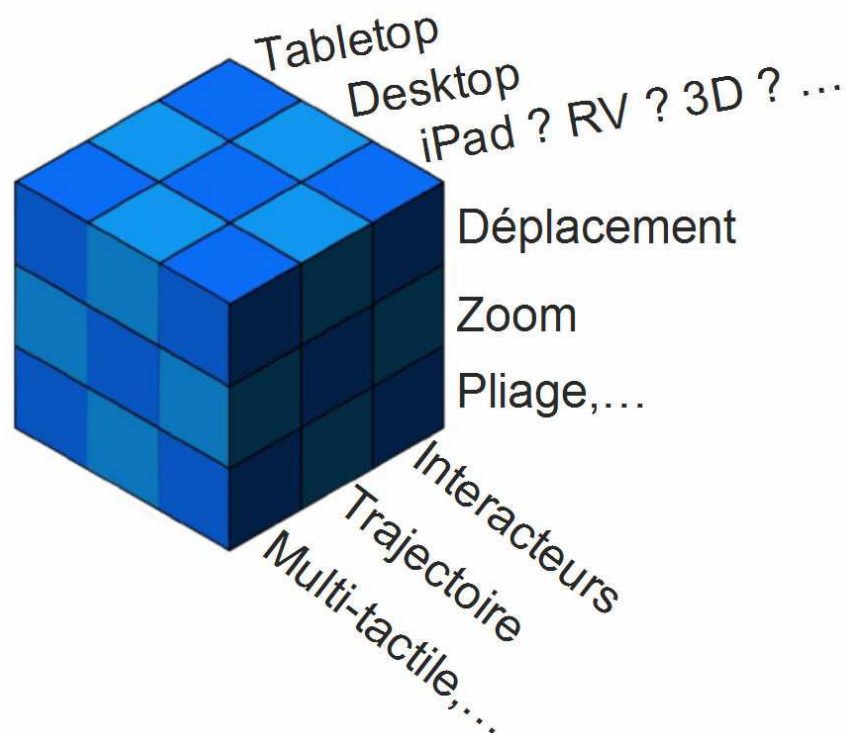


Figure 66 : Une matrice à trois dimensions peut facilement générer des dizaines de combinaisons. Une troisième dimension pour les dispositifs d'interaction permettrait d'étudier, par exemple, l'utilisation d'interacteurs pour le zoom sur plusieurs dispositifs.

Conclusion _____

L'objectif de cette thèse était d'envisager l'utilisation d'applications existantes mono-utilisateur par plusieurs utilisateurs autour d'une table interactive.

Nous avons mis en œuvre plusieurs solutions technologiques pour pouvoir utiliser les applications existantes, conçues pour des ordinateurs de bureau, avec un environnement de table interactive. En effet, les tables interactives ne sont pas des ordinateurs de bureau, elles ont des fonctionnalités différentes, et donc les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau. Nous avons présenté six technologies pour faire ce lien entre une application existante et un système de table interactive : la prise de capture d'écran à intervalle régulier, l'utilisation d'une carte graphique virtuelle ou d'un écran virtuel, la simulation d'un clavier virtuel et d'une souris virtuelle, l'utilisation d'un langage de script intégré à l'application ou au système d'exploitation, les interfaces de programmation dédiées à l'accessibilité numérique, et la modification d'une boîte à outils d'interface homme machine. Nous avons évalué ces technologies de lien selon six axes que nous avons précédemment introduits : la structuration des données, la flexibilité des modifications d'interface possibles, la performance, la compatibilité et la réutilisabilité, et la difficulté d'implémentation. La différence fondamentale entre ces technologies est la position du lien dans la chaîne de traitement entre l'application et le trio de dispositifs écran-clavier-souris pour lesquels l'application a été conçue. Nous avons trouvé que les technologies ont divers avantages et inconvénients, que nous avons résumés dans un tableau. La technologie « idéale » est probablement celle qui utilise tous les mécanismes que nous avons présentés, en fonction de l'application à utiliser sur la table interactive.

Ces technologies permettent de résoudre le problème posé par la différence entre l'interface de programmation d'applications (API) d'un ordinateur de bureau et l'API d'une table interactive. Mais elles ne résolvent pas les problèmes d'utilisabilité dont souffrent les applications conçues pour des ordinateurs quand elles sont transposées sur des tables interactives. En effet, les applications pour tables interactives ne se programment pas de la même manière que les applications pour ordinateurs de bureau, mais elles ne s'utilisent pas de la même façon non plus. Les interfaces utilisateur graphiques (GUI) des applications existantes ont été conçues pour être utilisées par un seul utilisateur avec un ordinateur de bureau. Pourtant, sur une même tâche réalisée collaborativement par deux utilisateurs sur une table interactive, l'utilisation d'une application pour ordinateurs de bureau non adaptée aux tables interactives est deux fois plus rapide, et jugée par 63% des utilisateurs plus facile à utiliser, qu'une application créée pour les tables interactives et prenant en compte les spécificités de celles-ci [21]. Les utilisateurs préfèrent également, à 53%, l'application plus lente et moins facile à utiliser, mais qui prend en compte les problèmes d'orientation autour de la table, de travail en parallèle, de gestion de la surface, et qui offre des techniques d'interaction adaptées aux conditions des tables interactives. Nous pensons que les logiciels existants, couplés à des techniques d'interaction

conçues en prenant en compte les spécificités des tables interactives, peuvent combiner les avantages des deux approches.

Nous avons présenté deux scénarios concrets où l'utilisation de logiciels existants sur une table interactive pose des problèmes d'utilisabilité. Nous remarquons que plusieurs de ces problèmes viennent des différences dans la gestion de l'espace sur les ordinateurs de bureau et les tables interactives. La nature de l'espace disponible sur la surface d'une table interactive et de l'espace disponible sur un écran d'ordinateur sont très différentes. Nous nous sommes donc intéressées à des techniques d'interaction pour la gestion de l'espace sur les tables interactives, qui sont compatibles avec les applications existantes. Nous voulons des techniques d'interaction qui puissent être intégrées de manière automatique et transparente pour l'application, à des applications existantes. Un exemple d'intégration automatique et transparente serait de remplacer les menus déroulants par des menus circulaires. Les fonctionnalités sont identiques et interchangeable. Le degré de flexibilité des modifications d'interface de la technologie utilisée pour amener l'application dans l'environnement de table interactive influe fortement sur les possibilités d'utiliser diverses techniques d'interaction.

Pour concevoir ces techniques d'interaction, nous avons introduit la métaphore du papier. Les métaphores sont utilisées, en interaction homme-machine, pour aider les utilisateurs à appréhender le fonctionnement d'un nouveau système plus rapidement en y appliquant des connaissances d'un système qu'ils connaissent déjà. Nous pensons qu'il est important de trouver de nouvelles sources d'inspiration pour concevoir des techniques d'interactions riches et naturelles pour les tables interactives. Le papier, probablement le support d'information le plus courant, offre de nombreuses possibilités d'interaction. Il peut être transporté facilement, rangé, classé, relié, découpé, stocké, annoté, effacé, ou au contraire archivé, il peut intégrer facilement du texte avec des images, graphiques, dessins, et combiner des parties imprimées et manuscrites. Nous proposons ainsi trois techniques d'interaction inspirées du papier. Nous proposons des fentes qui permettent de transmettre à distance des documents à un autre utilisateur, ainsi que d'effectuer des traitements sur les documents. Le pliage durable permet de nouvelles interactions avec un document. Il rend accessible le dos du document, qui peut contenir une interface graphique pour l'utilisateur. En observant le recto de documents papier existants, nous sommes arrivés à la conclusion que certains artefacts informatiques existants auraient toute leur place au dos d'un document : le système d'aide de l'application, des outils pour gérer le cycle de vie du document, des métadonnées sur le document,... Enfin, le plier pour empiler permet la création de piles de façon spontanée, en « recyclant » un document existant comme couverture pour de pile, simplement en le pliant en deux.

Certaines des interactions que nous avons proposées peuvent être effectuées de plusieurs manières. Ceci nous a inspiré la création d'une matrice des interactions dédiée aux interactions de gestion des documents sur la surface d'une table interactive. Nous nous basons sur les interactions déjà existantes dans la gestion des

fenêtres sur les ordinateurs de bureau : le déplacement et le redimensionnement, auxquelles nous ajoutons trois interactions : la rotation, le zoom et le pliage. Ces interactions s'approchent de devenir des standards en matière d'interface pour table interactive dans la communauté des *tabletops*. Nous avons identifié trois moyens distincts d'effectuer ces cinq interactions : l'utilisation d'un interacteur dédié, l'utilisation de la position du document sur la table, et l'utilisation de la trajectoire du doigt de l'utilisateur. Notre matrice met en évidence que ces trois moyens peuvent être proposés simultanément par le système informatique, imitant ainsi cette propriété du monde physique. Nous supposons que d'utiliser des moyens similaires pour effectuer plusieurs actions doit permettre aux utilisateurs de transférer latéralement les connaissances d'une technique d'interaction pour table interactive pour découvrir une autre technique d'interaction, de la même façon que l'utilisation d'une métaphore permet aux utilisateurs d'appliquer leurs connaissances du monde physique aux interactions avec les artefacts numériques.

Perspectives

Nous prévoyons d'étendre cette matrice à d'autres actions et d'autres moyens de les effectuer. A mesure que nous considérons d'autres actions et d'autres moyens, certains couples action-moyen vont ne correspondre à aucune technique d'interaction existante. La survenance de « cases vides » dans la matrice peut inspirer le concepteur de techniques d'interaction pour générer de nouvelles techniques en combinant des actions et des moyens existants. Nous envisageons également d'ajouter une troisième dimension à la matrice pour les dispositifs d'interaction : ordinateurs de bureau, tables interactives, périphériques portables,... A mesure que les dispositifs auxquels a accès l'utilisateur se démultiplient, le chercheur en interaction homme-machine peut vouloir comparer comment des actions et des moyens sont mis en œuvre sur les divers dispositifs interactifs. A nouveau, il est logique de penser que l'utilisateur d'un nouveau type de dispositifs peut se baser sur ses connaissances du fonctionnement de dispositifs similaires pour découvrir les techniques d'interaction.

Nos travaux ont touché à plusieurs facettes de l'interaction homme-machine (IHM). Ils concernent bien entendu l'étude de la nouvelle « forme » de systèmes informatiques que sont les tables interactives ou « *tabletop* ». Les systèmes *tabletop* prennent la forme d'une table dans son entier, qui réunit les fonctions de l'écran, du clavier et de la souris des ordinateurs de bureau en un seul périphérique partagé par plusieurs utilisateurs. Les tables interactives étant des ordinateurs intrinsèquement multi-utilisateur, ils concernent également le travail coopératif assisté par ordinateur (CSCW), spécifiquement l'interaction face-à-face colocalisée dans la matrice temps-espace des systèmes coopératifs [33]. Plusieurs études sur l'aspect coopératif des tables interactives indiquent que les protocoles sociaux qui sont mis en œuvre entre les utilisateurs colocalisés peuvent servir à résoudre les conflits d'interactions. Par exemple, la disposition spatiale des utilisateurs autour d'une même table apporte naturellement des mécanismes de répartition de l'espace [105]. Les utilisateurs sont également conscients du travail et des actions des autres utilisateurs de la table [87]. D'autres études notent que ces protocoles sociaux ne sont pas suffisants. Certains des problèmes relevés par [78], par exemple, nous semblent venir d'un manque de maturité dans la conception des applications pour tables interactives étudiées (e.g. une « vue » globale que n'importe quel utilisateur peut changer) ou des techniques d'interaction utilisées (e.g. une fonction « *magnet* » pour orienter tous les documents vers un utilisateur donné). Ce manque de maturité dans les applications ad-hoc développées par les chercheurs pourrait aussi expliquer certains des résultats de l'étude [21]. Nous envisageons de procéder à des études similaires en utilisant des applications existantes, dont la maturité des interfaces utilisateur graphiques s'est développée sur des dizaines d'années, pour isoler cet effet et étudier plus précisément l'impact de la table interactive en temps que dispositif d'interaction homme-machine.

En partenariat avec l'ENSAM, nous avons déjà mené une étude sur la comparaison, lors d'expériences contrôlées, entre l'utilisation de papier et de fournitures de bureau

physiques, d'une table interactive simulant ce papier et ces fournitures, et d'une table interactive offrant une application informatique complète [12, 13]. Les tâches retenues étaient des tâches de créativité, qui font l'objet de recherche dans le domaine de la psychologie et y sont définies de manière très précise. La condition de la table interactive simulant le papier et les fournitures du bureau utilisait exactement le même processus que la condition utilisant le papier et les fournitures de bureau physique. La condition utilisant une application informatique complète introduisait un paradigme différent, similaire aux différences entre les deux applications de [21], avec par exemple un accès à Internet lors de la séance de créativité. Les résultats montrent un avantage pour la table interactive simulant le papier et les fournitures du bureau. Elle a été jugée plus agréable par les utilisateurs, et rend la communication entre utilisateurs plus plaisante, toutefois sans augmenter la productivité globale du groupe. En revanche elle a permis de réduire significativement le phénomène négatif de paresse sociale (*social loafing*) qui se produit naturellement au sein d'un groupe de personnes dans ces situations de travail collaboratif. La paresse sociale se traduit par un déséquilibre dans la quantité de contribution des membres du groupe, et l'apparition de « leaders » et de « suiveurs ». Nous avons observé, lors de l'utilisation la table interactive simulant le papier et les fournitures du bureau, un rééquilibrage des contributions entre les membres du groupe. Cette observation pourrait être une conséquence de l'agréabilité de la communication entre utilisateurs, qui stimulerait la créativité en favorisant l'échange d'idées. En revanche, l'utilisation de l'application informatique complète sur la table interactive, dont nous admettons volontiers qu'elle puisse avoir souffert d'un manque de maturité et de cohérence dans les interactions proposées, a produit une diminution de la performance du groupe. La transposition de cette expérience aux conditions utilisation d'une application sur un ordinateur de bureau, utilisation de la même application sur une table interactive et utilisation d'une application spécifiquement développée pour une table interactive pourrait donner des résultats intéressants.

Les applications existantes pour ordinateurs de bureau sont celles que les utilisateurs utilisent dans leurs vies de tous les jours, que ce soit dans un contexte professionnel, personnel, pour les loisirs, etc. Leurs données sont stockées dans ces applications, leurs processus sont conçus autour de ces applications. L'utilisation de ces applications existantes sur des tables interactives permet d'envisager des études in situ à long terme de l'interaction avec une table numérique interactive. A cause du manque d'interopérabilité entre les tables interactives et les applications et les données existantes, la seule étude de terrain à notre connaissance [125] n'a porté que sur un seul utilisateur, utilisant une table interactive presque comme un ordinateur de bureau.

A plus long terme

L'utilisation d'applications existantes, avec leurs données et les processus qui les utilisent, peut permettre de faire entrer la table interactive dans le monde de l'entreprise. Les tables interactives sont déjà présentes dans les œuvres de fiction grand public, comme les séries télévisées et les films. Les tables interactives de fiction ne sont plus limitées à la science-fiction (comme *The Island* (figure 67 à gauche)), mais également présentes dans des œuvres se voulant réalistes, et dont l'action se situe à notre époque (comme *CSI: Miami* (figure 67 à droite), qui utilise une table qui serait une Microsoft Surface).

On entend souvent dire qu'il ne manque qu'une *killer app* aux tables interactives pour s'imposer dans le monde réel. Nous pensons qu'il n'y aura pas juste une *killer app*, juste un système de cartographie ou de partage de photos, mais que le déclic viendra de la capacité à s'interfacer avec l'existant. La *killer app*, finalement, c'est l'ensemble de toutes les *killer apps* des ordinateurs de bureau, disponibles sur les tables.

Nous espérons ensuite voir le développement d'un écosystème conséquent de tables interactives, qui puisse agir comme un démarreur pour la création de nouvelles applications professionnellement conçues, multi-touche, multi-utilisateur, spécifiquement développées pour les *tabletops* et tirant partie de toutes les possibilités d'interaction collaborative qu'elles offrent. Et je veux faire partie de cette aventure.



Figure 67 : Des tables interactives de fictions : dans le film de science-fiction *The Island* (à gauche) et la série télévisée réaliste *CSI: Miami* (à droite). [Images soumises aux droits d'auteur, utilisées à titre d'illustration.]

Bibliographie

- [1] Abdel-Wahab, H., Peit, M. XTV: A framework for sharing X Window clients in remote synchronous collaboration. *Proceedings of IEEE Tricomm Conference (1991)*, pp. 159–167.
- [2] Agarawala, A., Balakrishnan, R. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. *Proceedings of CHI 2006*, pp. 1283–1292.
- [3] Apple Inc. AppleScript: The Language of Automation. <http://www.apple.com/applescript/>
- [4] Baecker, R., Grudin, J., Buxton, W., Greenberg, S. *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann Publishers, 1995.
- [5] Bartis, S., Szymanski, K., Harkins, S. Evaluation and performance: A two-edged knife. *Personality and Social Psychology Bulletin*, Vol. 14 (1988), pp. 242–251.
- [6] Beaudouin-Lafon, M. Novel Interaction Techniques for Overlapping Windows. *Proceedings of UIST 2001*, pp. 153–154.
- [7] Bentley, R., Dourish, P. Medium versus mechanism: Supporting collaboration through customisation. *Proceedings of ECSCW'95*, pp. 133–148.
- [8] Berard, F. The Magic Table: Computer Vision Based Augmentation of a Whiteboard for Creative Meetings. *Proceedings of PROCAMS'03: IEEE International Conference in Computer Vision, Workshop on Projector-Camera Systems*.
- [9] Berry, L. Bartram, L., Booth, K. Role-based control of shared application views. *Proceedings of UIST'05*, pp. 23–32.
- [10] Besacier, G., Rey, G., Najm, M., Buisine, S., Vernier F. Paper Metaphor for Tabletop Interaction Design. *Proceedings of HCI International 2007*
- [11] Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T. Toolglass and Magic Lenses: The See-Through Interface. *Proceedings of Siggraph'93*, pp. 73–80.
- [12] Buisine, S., Besacier, G., Aoussat, A., Vernier, F. How do interactive tabletop systems influence collaboration? To appear in *International Journal of Human-Computer Studies*. Elsevier.
- [13] Buisine, S., Besacier, G., Najm, N., Aoussat, A., Vernier, F. Computer-supported creativity: Evaluation of a tabletop mind-map application. *Proceedings of 7th international conference on Engineering psychology and cognitive ergonomics (2007)*, pp. 22–31.
- [14] Buxton, W., Myers, B. A study in two-handed input. *Proceedings of CHI'86*, pp. 321–326.
- [15] Chapuis, O., Roussel, N. Metisse is not a 3D desktop! *Proceedings of UIST'05*, pp. 13–22.
- [16] Chapuis, O., Blanch, R., Beaudouin-Lafon, M. Fitts' Law in the Wild: A Field Study of Aimed Movements. LRI Technical Report 1480, Laboratoire de Recherche en Informatique (2007).
- [17] Chapuis, O., Roussel, N. Copy-and-paste between overlapping windows. *Proceedings of CHI'07*, pp. 201–210.
- [18] Chatty, S. Extending a graphical toolkit for two-handed interaction. *Proceedings of UIST'94*, pp. 195–204.
- [19] Chatty, S., Lecoanet, P. Un poste de travail avec reconnaissance de gestes pour le contrôle aérien. *Actes de IHM'95*, pp. 81–88.
- [20] Collins, A., Apted T., and Kay J. Tabletop file system access: Associative and hierarchical approaches. *Proceedings of Tabletop 2007*, pp. 113–120.
- [21] Collins, A., Bezerianos, A., McEwan, G., Rittenbruch, M., Wasinger, R., Kay, J. Understanding file access mechanisms for embedded Ubicomp collaboration interfaces. *Proceedings of Ubicomp'09*, pp. 135–144.
- [22] Crowley, T., Milazzo, P., Baker, E., Forsdick, H., Tomlinson, R. MMConf: An infrastructure for building shared multimedia applications. *Proceedings of CSCW'90*. pp. 329–342.
- [23] Denoue, L., Nelson, L., Churchill, E. A fast, interactive 3D paper-flier metaphor for digital bulletin boards. *Proceedings of UIST'03*, pp. 169–172.

- [24] Dietz, P., Leigh, D. DiamondTouch: A Multi-User Touch Technology. Proceedings of UIST'01, pp. 219–226.
- [25] Dragicevic, P. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. Proceedings of UIST'04, pp. 193–196.
- [26] Duffy, T., Mehlenacher, B., Palmer, J. On Line Help: Design and Evaluation. Ablex (Norwood, NJ). 1992.
- [27] Dugosh, K., Paulus, P. Cognitive and social comparison processes in brainstorming. Journal of Experimental Social Psychology, Volume 41 (2005), pp. 313–320.
- [28] Dugosh, K., Paulus, P., Roland, E., Yang, H. Cognitive stimulation in brainstorming. Journal of Personality and Social Psychology, Volume 79, Issue 5 (2000), pp. 722–735.
- [29] Eckstein, R., Loy, M., Wood, D. Java Swing. O'Reilly & Associates, Inc., 1998.
- [30] Edwards, W. Flexible conflict detection and management in collaborative applications. Proceedings of UIST '97, pp. 139–148.
- [31] Edwards, W., Hudson, S., Rodenstein, R., Smith, I., Rodrigues, T. Systematic output modification in a 2D UI Toolkit. Proceedings of UIST'97, pp. 151–158.
- [32] Edwards, W., Mynatt, E. Timewarp: techniques for autonomous collaboration. Proceedings of CHI'97, pp. 218–225.
- [33] Ellis, C., Gibbs, S., Rein, G. Groupware: some issues and experiences. Communications of the ACM, Volume 34, Issue 1 (Jan. 1991), pp. 39–58.
- [34] Engelbart, D. NLS teleconferencing features: The Journal, and Shared-Screen Telephoning. Proceedings of IEEE Fall COMPCON (1975), pp. 173–176.
- [35] Epps, J., Lichman, S., Wu, M. A study of hand shape use in tabletop gesture interaction. CHI'06 Extended Abstracts, pp. 748–753.
- [36] European Computer Manufacturers Association. Standard ECMA-234, Application Programming Interface for Windows (APIW), December 1995.
- [37] Forlines, C., Shen, C., Buxton, B. Glimpse: a novel input model for multi-level devices. Ext. Abstracts CHI 2005, pp. 1375–1378.
- [38] Forsdick, H. Explorations in real-time multimedia conferencing. Proceedings of 2nd International Symposium on Computer Message Systems (1985), pp. 299–315.
- [39] Furuichi, M., Mihori, Y., Muraoka, F., Esenther, A., Ryall, K. DTMap Demo: Interactive Tabletop Maps for Ubiquitous Computing. Proceedings of UbiComp 2005.
- [40] Garfinkel, D., Welti, B., Yip, T. HP Shared X: A tool for real-time collaboration. Hewlett-Packard Journal Vol. 45 (2), April 1994, pp. 23–36.
- [41] Hancock, M., Vernier, F., Wigdor, D., Carpendale, S., Shen, C. Rotation and Translation Mechanisms for Tabletop Interaction. Proceedings of Tabletop'06, pp. 79–88.
- [42] Harkins, S., Jackson, J. The role of evaluation in eliminating social loafing. Personality and Social Psychology Bulletin, Vol. 11, Issue 4 (1985), pp. 457–465.
- [43] Hartmann B., Morris M., Cassanego A. Reducing clutter on tabletop groupware systems with tangible drawers. Adj. Proceedings of UbiComp 2006.
- [44] Hinrichs, U., Carpendale, S., Scott, S., Pattison, E. Interface Currents: Supporting fluent collaboration on tabletop displays. Proceedings of Smart Graphics'05, pp. 185–197.
- [45] Ho, W., Olsson, A. An approach to genuine dynamic linking. Software – Practice & Experience, Vol. 21, Issue 4 (Apr. 1991), pp. 375–390.
- [46] Holman, D., Vertegaal, R., Altosaar, M., Troje, N., Johns, D. Paper windows: interaction techniques for digital paper. Proceedings of CHI'05, pp. 591–599.
- [47] Hong, L., Card, S., and Chen, J. Turning Pages of 3D Electronic Books. Proceedings of 3DUI 2006, pp. 159–165.

- [48] Huang, E., Mynatt, E. Semi-public displays for small, co-located groups. *Proceedings of CHI 2003*, pp. 49–56.
- [49] Hudson, S., Smith, I. SubArctic UI toolkit user's manual. Technical report, College of Computing, Georgia Institute of Technology, 1996.
- [50] Hunt, G., Brubacher, D. Detours: Binary Interception of Win32 Functions. *Proceedings of 3rd USENIX Windows NT Symposium*, pp. 135–143.
- [51] Hutchings, D., Smith, G., Meyers, B., Czerwinski, M., Robertson, G. Display space usage and window management operation comparisons between single monitor and multiple monitor users. *Proceedings of AVI'04*, pp. 32–39.
- [52] Hutchings, D., Stasko, J. Shrinking window operations for expanding display space. *Proceedings of AVI'04*, pp. 350–353.
- [53] Hutterer, P., Close, B., Thomas, B. Supporting Mixed Presence Groupware in Tabletop Applications. *Proceedings of Tabletop 2006*, pp. 63–70.
- [54] Hutterer, P., Thomas, B. Groupware support in the windowing system. *Proceedings of 2007 Australasian Conference on User interface*, Vol. 64, pp. 39–46.
- [55] Inkpen, K., Booth, K., Gribble, S., Klawe, M. Give and take: children collaborating on one computer. *Proceedings of CHI'95*, pp. 258–259.
- [56] Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M. Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. *Proceedings of UIST 2003*, pp. 159–168.
- [57] Kacmar, C. A process approach to providing hypermedia services to existing non-hypermedia applications. *Electronic Publishing: Organization, Dissemination, and Distribution*, Vol. 8(1), March 1995, pp. 31–48.
- [58] Kakehi, Y., Iida, M., Naemura, T., Shirai, Y., Matsushita, M., Ohguro, T. Lumisight Table: Interactive View-Dependent Tabletop Display Surrounded by Multiple Users. *IEEE Computer Graphics and Applications*, 25(1):48–53, January 2005.
- [59] Kay, A. User Interface: A Personal View. In *The Art of Human-Computer Interface Design*, B. Laurel, ed., Addison-Wesley Publishing Company, 1990, pp. 191–207.
- [60] Kearsley G. *Online Help Systems: Design and Implementation*. Ablex (Norwood, NJ). 1988.
- [61] Kruger, R., Carpendale, S., Scott, S., Tang, A. Fluid integration of rotation and translation. *Proceedings of CHI'05*, pp. 601–610.
- [62] Lantz, K. An experiment in integrating multimedia conferencing. *Proceedings of CSCW'86*, pp. 267–275.
- [63] Lauwers, J., Joseph, T., Lantz, K., Romanow, A. Replicated architectures for shared window systems: A critique. *Proceedings of ACM Conference on Organization Information Systems (1990)*, pp. 249–260.
- [64] Lauwers, J., Lanta, K. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. *Proceedings of CHI'90*, pp. 303–311.
- [65] Lecolinet, E. Pointeurs multiples : étude et implémentation. *Actes d'IHM 2003*, pp. 134–141.
- [66] Liu, J., Pinelle, D., Sallam, S., Subramanian, S., Gutwin, C. TNT: improved rotation and translation on digital tables. *Proceedings of Graphics interface 2006*, pp. 25–32.
- [67] Machin, A., *The Educational Origins of Tourism*. Tourism, Hospitality and Events School, Leeds Metropolitan University. <http://text.alanmachinwork.net/The-Educational-Origins-of-Tourism>
- [68] MacIntyre, B., Mynatt, E., Volda, S., Hansen, K., Tullio, J., Corso, G. Support for multitasking and background awareness using interactive peripheral displays. *Proceedings of UIST'01*, pp. 41–50.

- [69] Mackay, W. Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers. *Proceedings of AVI'98*, pp. 13–21.
- [70] MacKenzie, I., Ware, C. Lag as a determinant of human performance in interactive systems. *Proceedings of INTERACT'93 and CHI'93*, pp. 488–493.
- [71] Malone, T. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Inf. Syst.* 1, 1, Jan. 1983, pp. 99–112.
- [72] Mander, R., Salomon, G., Wong, Y. A “Pile” Metaphor for Supporting Casual Organization of Information. *Proceedings of CHI'92*, pp. 627–634.
- [73] Matsushita, M., Iida, M., Ohguro, T., Shirai, Y., Kakehi, Y., Naemura, T. Lumisight table: a face-to-face collaboration support system that optimizes direction of projected information to each stakeholder. *Proceedings of CSCW'04*, pp. 274–283.
- [74] Mertz, C., Chatty, S., Vinot, J. Pushing the limits of ATC user interface design beyond S&M interaction: the DigiStrips experience. 3rd USA/Europe Air Traffic Management R&D Seminar (2000).
- [75] Michinov, N., Primois, C. Improving productivity and creativity in online groups through social comparison process: New evidence for asynchronous electronic brainstorming. *Computers in Human Behavior*, Vol. 21 (2005), pp. 11–28.
- [76] Montferrat, P., Apostolopoulou, A., Besacier, G., Buisine, S., Rey, G., Vernier, F. Brainstorming and Mind-Mapping on DiamondTouch: Creativity Unleashed. 2nd DiamondTouch Workshop, from Personal Computer to Multi-user Collaboration, 5 June 2007, Amsterdam, the Netherlands.
- [77] Morris, M., Huang, A., Paepcke, A., Winograd, T. Cooperative gestures: multi-user gestural interactions for co-located groupware. *Proceedings of CHI'06*, pp. 1201–1210.
- [78] Morris, M., Ryall, K., Shen, C., Forlines, C., Vernier, F. Beyond “Social Protocols”: Multi-User Coordination Policies for Co-located Groupware. *Proceedings of CSCW 2004*, pp. 262–265.
- [79] Myers, B. An Implementation Architecture to Support Single-Display Groupware. Carnegie Mellon University School of Computer Science Technical Report, CMU-CS-99-139 and Human Computer Interaction Institute Technical Report CMU-HCII-99-101, May, 1999.
- [80] Myers, B., Peck, C., Nichols, J., Kong, D., Miller, R. Interacting At a Distance Using Semantic Snarfing. *Proceedings of UbiComp'2001* pp. 305–314.
- [81] Myers, D., Bazinet, A. Intercepting arbitrary functions on Windows, UNIX, and Macintosh OS X platforms. Technical Report CS-TR-4585, UMIACS-TR-2004-28, University of Maryland (2004).
- [82] Mynatt, E., Edwards, W. Mapping GUIs to auditory interfaces. *Proceedings of UIST'92*, pp. 61–70.
- [83] Najm, M., Besacier, G., Buisine, S., Vernier, F. La réalité augmentée dans le cadre du travail collaboratif. *CONFERE'06 Colloque francophone sur les sciences de l'innovation*.
- [84] Niederauer, C., Houston, M., Agrawala, M., Humphreys, G. Non-invasive interactive visualization of dynamic architectural environments. *Proceedings of I3D'03*, pp 55–58.
- [85] Nijstad, B., Stroebe, W., Lodewijks, H. Cognitive stimulation and interference in groups: Exposure effects in an idea generation task. *Journal of Experimental Social Psychology*, Vol. 38 (2002), pp. 535–544.
- [86] Parente, P., Clippingdale, B. Linux screen reader: extensible assistive technology. *Proceedings of Assets'06*, pp. 261–262.
- [87] Pauchet, A., Coldefy, F., Lefebvre, L., Louis Dit Picard, S., Perron, L., Bouguet, A., Collobert, M., Guerin, J., Corvaisier, D. TableTops: worthwhile experiences of collocated and remote collaboration. *Proceedings of Tabletop'07*, pp. 27–34.
- [88] Paulus, P., Dzindolet, M. Social influence processes in group brainstorming. *Journal of Personality and Social Psychology*, Vol. 64, Issue 4 (1993), pp. 575–586.

- [89] Peeling, N., Satchell, J. Analysis of the Impact of Open-Source Software. QinetiQ Ltd. QINETIQ/KI/SEB/CR010223 (2001).
- [90] "Portable Executable win32 API Compatible Environment (PEACE)" project.
<http://chiharu.haun.org/peace/status.html>
- [91] Pratchett, T., Briggs, S. Libraries, nature of. *The Discworld Companion*, 1997, éditeur Gollancz.
- [92] Richardson, T., Stafford-Fraser, Q., Wood, K., Hopper, A. Virtual Network Computing. *IEEE Internet Computing*, Vol. 2, No. 1 (1998), pp. 33–38.
- [93] Ringel, M., Ryall, K., Shen, C., Forlines, C., Vernier, F. Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. *Proceedings of CHI'04, Extended Abstracts*, pp. 1441–1444.
- [94] Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., Smith, G. Scalable Fabric: flexible task management. *Proceedings of AVI'04*, pp. 85–89.
- [95] Robinson, J., Robertson, C. The LivePaper System: Augmenting Paper on an Enhanced Tabletop. *Computers & Graphics*, Vol. 25, Issue 5 (2001), pp. 731–743.
- [96] Rogers, Y., Hazlewood, W., Blevis, E., Lim, Y. Finger talk: collaborative decision-making using talk and fingertip interaction around a tabletop display. *CHI'04 Ext. Abst.*, pp. 1271–1274.
- [97] Rogers, Y., Rodden, T. Configuring spaces and surfaces to support collaborative interactions. In O'Hara, K., Perry, M., Churchill, E., Russell, D. (éditeurs) *Public and Situated Displays*. Kluwer Publishers (2004), pp. 45–79.
- [98] Roussel, N. Ametista: a mini-toolkit for exploring new window management techniques. *Proceedings of the Latin American Conference on Human-Computer interaction: CLIHC'03*, vol. 46, pp. 117–124.
- [99] Roussel, N., Chapuis, O. Metisse : un système de fenêtrage hautement configurable et utilisable au quotidien. *Actes de IHM 2005*.
- [100] Ryall, K., Forlines, C., Shen, C., Morris, M. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. *Proceedings of CSCW'04*, pp. 284–293.
- [101] Scheifler, R., Gettys, J. The X window system. *ACM Transactions on Graphics (TOG)*, Volume 5, Issue 2 (April 1986), pp. 79–109.
- [102] Scott, S., Carpendale, S., Habelski, S. Storage bins: Mobile storage for collaborative tabletop displays. In *IEEE Comp. Graphics and Applications*, Vol. 25, Issue 4, 2005, pp. 58–65.
- [103] Scott, S., Carpendale, S., Inkpen, K. Territoriality in collaborative tabletop workspaces. *Proceedings of CSCW'04*, pp. 294–303.
- [104] Scott, S., Grant, K., Mandryk, R. System Guidelines for Colocated, Collaborative Work on a Tabletop Display. *Proceedings of ECSCW'03*, pp. 159–178.
- [105] Scott, S. Territoriality in Collaborative Tabletop Workspaces. Dissertation for the degree of doctor of philosophy, Univ. of Calgary, 2005.
- [106] Sellen, A., Harper, R. *The Myth of the Paperless Office*, MIT Press, 2001.
- [107] Shen, C., Ryall, K., Forlines, C., Esenther, A., Vernier, F., Everitt, K., Wu, M., Wigdor, D., Morris, M., Hancock, M., Tse, E. Informing the Design of Direct-Touch Tabletops. *IEEE Comput. Graph. Appl.*, Vol. 26, Issue 5 (Sep. 2006), pp. 36–46.
- [108] Shen, C., Vernier, F., Forlines, C., Ringel, M. DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction. *Proceedings of CHI 2004*, pp. 167–174.
- [108b] Shen, C. Interactive tabletops, User Interface, Metaphors and Gestures. *ACM SIGGRAPH 2007 Courses*, pp. 40–53.

- [109] Shipman, F., Marshall, C., Moran, T. Finding and using implicit structure in human-organized spatial layouts of information. *Proceedings of CHI'95*, pp. 346–353.
- [110] Sire, S., Chatty, S., Gaspard-Bouline, H., Colin F. How can groupware preserve our coordination skills? Designing for direct collaboration. *Proceedings of Interact 99*, pp. 304–312.
- [111] Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., Suchman, L. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (Jan. 1987), pp. 32–47.
- [112] Sternberg, R., Kaufman, J., Pretz, J. The creativity conundrum: A propulsion model of creative contributions. Philadelphia (2002).
- [113] Stewart, J., Bederson, B., Druin, A. Single display groupware: a model for co-present collaboration. *Proceedings of CHI'99*, pp. 286–293.
- [114] Stuerzlinger, W., Chapuis, O., Phillips, D., Roussel, N. User Interface Façades: Towards Fully Adaptable User Interfaces. *Proceedings of UIST'06*, pp. 309–318.
- [115] Swan, L., Taylor, A., Harper, R. Making place for clutter and other ideas of home. *ACM TOCHI* Vol. 15, Issue 2 (Jul. 2008), pp. 1–24.
- [116] Tang, A., Tory, M., Po, B., Neumann, P., Carpendale, S. Collaborative coupling over tabletop displays. *Proceedings of CHI'06*, pp. 1181–1190.
- [117] Thatcher, J. Screen reader/2: access to OS/2 and the graphical user interface. *Proceedings of Assets'94*, pp. 39–46.
- [118] Tse, E., Greenberg, S., Shen, C., Barnwell, J., Shipman, S. and Leigh, D. Multimodal Split View Tabletop Interaction Over Existing Applications. *Proceedings of Tabletop 2007*
- [119] Tsichritzis, D. Form management. *Com. ACM* 25, 7 (July 1982), pp. 453–478.
- [120] Tuddenham, P., Robinson, P. T3: Rapid Prototyping of High-Resolution and Mixed-Presence Tabletop Applications. *Proceedings of Tabletop 2007*, pp. 11–18.
- [121] Ullmer, B., Ishii, H. Emerging Frameworks for Tangible User Interfaces. *IBM Systems Journal*, Vol 39, Issue 3–4 (July 2000), pp. 915–931.
- [122] van Dam, A. Post-WIMP user interfaces. *Communications of the ACM*, Volume 40, Issue 2 (February 1997)
- [123] Vernier, F., Lesh, N., Shen, C. Visualization techniques for circular tabletop interfaces. *Proceedings of AVI 2002*, pp. 257–265.
- [124] Westergaard, M. Supporting Multiple Pointing Devices in Microsoft Windows. In *Microsoft Summer Workshop for Faculty and PhDs*, Cambridge, England, September 2002.
- [125] Wigdor, D., Perm, G., Ryall, K., Esenther, A., Shen, C. Living with a Tabletop: Analysis and Observations of Long Term Office Use of a Multi-Touch Table. *Proceedings of Tabletop'07*, pp 60–67.
- [126] Wigdor, D., Shen, C., Forlines, C., Balakrishnan, R. Effects of display position and control space orientation on user preference and performance. *Proceedings of CHI'06*, pp. 309–318.
- [127] Williamson, C. Shneiderman, B. The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system. *Proceedings of SIGIR'92*, pp. 338–346.
- [128] “Wine Is Not an Emulator” project. <http://www.winehq.org/about/>
- [129] Wu, M., Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *Proceedings of UIST'03*, pp. 193–202.
- [130] Xia, S., Sun, D., Sun, C., Chen, D., Shen, H.F. Leveraging single-user applications for multi-user collaboration: the CoWord approach. *Proceedings of CSCW 2004*, pp. 162–171.

Index des figures _____

Figure 1 : La table interactive Lumisight	14
Figure 2 : Le Xerox Alto	16
Figure 3 : Le composant standard « bouton »	17
Figure 4 : Schéma du fonctionnement d'un système d'exploitation de bureau et des façons dont un système de table interactive peut venir se greffer sur ses liens de communication.....	21
Figure 5 : Architecture de notre implémentation du système de graphique virtuelle	27
Figure 6 : Plusieurs applications Linux existantes utilisées sur une table interactive	28
Figure 7 : Architecture de notre implémentation du système basé sur la technologie de scripting Automation, permettant d'utiliser l'application existante Microsoft Excel avec une table interactive	32
Figure 8 : Une boîte de dialogue de l'application existante Microsoft Excel, et une partie des informations d'accessibilité fournies par le logiciel <i>Accessibility Explorer tool</i>	33
Figure 9 : Architecture du système utilisant des modifications d'une boîte à outils d'interface utilisateur existante	37
Figure 10 : Les fonctions de l'interface de programmation d'application (API) win32 concernant aux menus, et la façon dont elles interagissent entre elles et avec les structures de données de menus	43
Figure 11 : Architecture de notre implémentation du système utilisant des modifications d'une boîte à outils d'interface utilisateur win32	47
Figure 12 : Technologies permettant d'utiliser une application existante avec un environnement différent du point de vue de son modèle de sortie graphique et d'entrée utilisateur	52
Figure 13 : Deux logiciels de gestion de fichiers utilisés sur une table interactive : le logiciel existant Windows Explorer, et le logiciel pour table interactive Focus. Le logiciel existant est plus rapide et plus facile à utiliser	56
Figure 14 : l'application existante GIMP, utilisée sur un ordinateur de bureau	58
Figure 15 : l'application existante GIMP utilisée par trois utilisateurs sur une table interactive	59
Figure 16 : Le formulaire du scénario 2, ouvert dans le logiciel Microsoft Excel, dans la configuration classique d'un ordinateur de bureau	60
Figure 17 : Le logiciel pour ordinateur de bureau Microsoft Excel, utilisé par quatre utilisateurs sur une table interactive	61
Figure 18 : Le projet <i>Tangible Drawers</i> : des « tiroirs de bureau » physiques « contiennent » des documents numériques	68
Figure 19 : Un <i>junk drawer</i> , un exemple d'utilisation spontanée des métaphores par les utilisateurs pour reproduire leurs usages existants avec les objets inspirant la métaphore	69
Figure 20 : Publicité pour le Xerox Star (1981), le contenu d'un document numérique affiché à l'écran est comparé avec un document imprimé sur du papier	70
Figure 21 : Le projet <i>LivePaper</i> : des feuilles de papiers ordinaires augmentées avec des informations numériques.....	71
Figure 22 : Le projet <i>PaperWindows</i> : des fenêtres d'applications transférées sur des feuilles de papiers physiques avec lesquelles l'utilisateur peut interagir au doigt ou au stylo.....	72
Figure 23 : Un livre numérique ouvert à une double page	73
Figure 24 : Un « strip » utilisé dans le contrôle aérien en version papier et numérique	73
Figure 25 : Du papier virtuel au rendu réaliste	74
Figure 26 : Des exemples de fentes dans lesquelles sont insérées des feuilles de papiers	75
Figure 27 : Une fente virtuelle pour transmettre des documents sur une table interactive.....	76
Figure 28 : Des exemples de feuilles de papier pliées	78

Figure 29 : Deux projets utilisant le pliage des fenêtres dans le cadre d'un ordinateur de bureau.....	79
Figure 30 : Une fenêtre pliée, dont le dos contient des éléments d'interface graphique utilisateur	81
Figure 31 : Une fenêtre, représentant un post-it virtuel, créée avec la boîte à outils DiamondSpin, utilisant des interacteurs placés à l'extérieur de la fenêtre.....	82
Figure 32 : L'utilisation des coins des fenêtres dans les environnements pour ordinateurs de bureau.....	83
Figure 33 : Diverses icônes de documents : toutes les icônes présentées ont leur coin supérieur droit plié.....	84
Figure 34 : L'interface graphique <i>BumpTop</i> , qui utilise un stylet et un moteur de simulation des règles de la physique pour interagir avec les icônes.....	86
Figure 35 : Répartition de l'interaction sur la table : le bord de la table est une zone privilégiée pour l'interaction, par rapport au centre de la table	87
Figure 36 : Trois façons de plier un document	88
Figure 37 : Des formulaires papier de l'administration française, dont le verso des formulaires contient des indications sur la façon de remplir le recto, ou plus généralement des informations sur le processus dont le formulaire fait partie.....	86
Figure 38 : Une application pour table interactive de dessin, utilisant le recto et le verso des fenêtres	91
Figure 39 : Des outils pour créer et modifier le contenu du document affichés au même plan que des outils pour gérer le document lui-même dans des applications pour ordinateurs de bureau.....	92
Figure 40 : Un post-it virtuel qui se plie automatiquement pour simuler une boîte de dialogues modale.....	93
Figure 41 : Des gestes coopératifs pour créer l'équivalent des boîtes de dialogues modales au niveau du système en entier.....	94
Figure 42 : Wikipédia, utilisant un système collaboratif de gestion de contenu de type « wiki », qui affiche le contenu et les méta-informations du contenu sur des pages distinctes.....	95
Figure 43 : Des photographies numériques dont le verso contient des informations sur la photographie et le sujet photographié	96
Figure 44 : Un document numérique composé de film de plastique virtuel	96
Figure 45 : Une pile de papiers maintenus par une feuille de papier pliée en deux.....	98
Figure 46 : Avant et après : création d'une pile numérique	99
Figure 47 : Une interface placée au dos de la couverture de la pile pour contrôler l'ensemble de la pile.....	101
Figure 48 : Un rolodex, un type de carnet d'adresses se présentant sous la forme d'un ensemble ordonné de cartes intercalées d'onglets pour accéder rapidement à une carte donnée	102
Figure 49 : Un grand document dans une petite pile	106
Figure 50 : Calcul de l'intersection d'un document actif avec une couverture de pile potentielle	108
Figure 51 : Machine à états du plier pour empiler.....	109
Figure 52 : Plusieurs façons d'effectuer l'interaction « pliage » avec du papier physique, toutes disponibles simultanément.....	115
Figure 53 : Les différentes actions que nous considérons.....	156
Figure 54 : Deux barres de titres pour éviter l'occlusion du contenu des fenêtres par la main de l'utilisateur lors de diverses situations de manipulation des fenêtres	118

Figure 55 : Matrice de l'utilisation d'un interacteur dédié.....	118
Figure 56 : L'environnement pour tables interactives Interface Currents.....	119
Figure 57 : Rotation autour de la position de l'utilisateur.....	121
Figure 58 : Utilisation de l'orientation des documents pour construire un arbre.....	122
Figure 59 : Matrice de l'utilisation de la position de la fenêtre pour modifier d'autres caractéristiques de celle-ci	123
Figure 60 : L'utilisation de forces de friction, combinée à une force de déplacement appliquée sur le bord du document, pour tourner le document selon la trajectoire du déplacement	125
Figure 61 : L'analogie avec une feuille de papier sur laquelle se base l'interaction TNT, qui permet de réorienter un document en le déplaçant	125
Figure 62 : Déplacement et orientation d'un document en un seul mouvement	126
Figure 63 : Matrice de l'utilisation de la trajectoire du doigt	127
Figure 64 : Une interaction bi-manuelle utilisant deux points de contrôle indépendants pour manipuler simultanément la position, l'échelle et l'orientation d'un document.....	128
Figure 65 : Différents moyens d'activer différentes interactions : la matrice des interactions que nous avons présentée dans cette thèse	130
Figure 66 : Une matrice à trois dimensions pouvant facilement générer des dizaines de combinaisons.....	131
Figure 67 : Des tables interactives de fictions : dans le film de science-fiction <i>The</i> et la série télévisée réaliste <i>CSI: Miami</i>	138