THÈSE
Présentée par
Céline Coutrix

Pour obtenir le titre de
DOCTEUR de L'UNIVERSITÉ JOSEPH FOURIER GRENOBLE 1
(arrêtés ministériel du 5 juillet 1984 et du 30 mars 1992)
Spécialité : Informatique

Interfaces de Réalité Mixte : Conception et Prototypage

Soutenue le 7 mai 2009 devant le jury composé de

Président	Philippe Lalanda
Directrice de thèse	Laurence Nigay
Rapporteurs	Wendy Mackay
	Pierre Leclercq
Examinateurs	Jean-Baptiste de la Rivière
	Philip Gray
	Emmanuel Dubois

Thèse préparée au sein du Laboratoire d'Informatique de Grenoble (LIG)
385, rue de la Bibliotheque, BP 53, 38041 GRENOBLE CEDEX 9
Université Joseph Fourier Grenoble 1

Remerciements

Je voudrais d'abord remercier les rapporteurs pour leurs commentaires précieux, ainsi que ceux qui ont participé directement au travail présenté ici : Laurence Nigay et toute l'équipe IIHM, pour leurs conseils avisés pendant la thèse et/ou le master et pour créer des conditions propices à la réussite d'une thèse.
Pour leur aide apportée dans les évaluations, je remercie Laurence Pasqualetti et Nadine Mandran, mais également tous les sujets, anonymes, qui ont gentiment accepté de faire don de leur temps.
Pour leur participation à la conception et à la réalisation des systèmes présentés, je remercie Philippe Renevier, Yilun You, Émilie Annweiler et l'équipe de Singapour, mais aussi Vincent Fréville, Jérôme Drouin, et Henri Derruder.

Je voudrais remercier aussi ceux qui m'ont permis de travailler en lien avec les arts plastiques, Martin Barré et surtout Sophie Mari, amis de longue date voire de bac à sable . Merci pour les discussions, les tuyaux, les conseils.
Je voudrais aussi remercier Samuel Bianchini, pour m'avoir offert la possibilité d'entrer à l'École Nationale Supérieure des Arts Décoratifs, et également tous les participants de l'EnsadLab DRII et FDM, encadrants et étudiants. Et je n'oublie pas Dan, qui m'a si souvent laissé squatter chez lui pour aller à l'ENSAD !

Et comme le travail n'est rien sans épanouissement personnel, je voudrais remercier Marc qui a été là pendant le Master Recherche, et surtout Sergi pour sa précieuse présence tout au long de la thèse : *Moltes gràcies Sergi per ser com ets.*
Merci à mes parents et mon frère Cyril, qui m'ont très justement parlé d'autre chose pendant ces années, sans oublier mes grands-parents très important . Ils m'ont sans doute fait dons de défauts héréditaires, mais aussi de qualités, d'ambition et de volonté, hautement nécessaires pour venir à bout de ce travail.
Mes amis et tous les gens dont j'ai apprécié la compagnie pendant ces années : des amis de longue date ou des copains de l'ENSIMAG partis trop tôt ☺, restés à Grenoble ou en thèse, en passant par les gens qui y sont arrivés entre temps, jusqu'aux copains/copines de copains/copines de copains/copines etc. avec qui j'ai bien rigolé. Je ne citerais pas de noms de peur d'en oublier, mais tout le monde est là. Merci d'avoir créer autour de moi un monde tellement chouette qu'aucune série américaine ne pourra jamais l'égaler, même pas *The Big Bang Theory* ou *Sex and the City*… ☺

Table des Matières

Chapitre 1 : Introduction 11
1. Constat 12
2. Objectifs 12
3. Contributions 13
3.1. Contribution conceptuelle 13
3.2. Contribution pratique 14
4. Structure du mémoire 14
4.1. Partie I : Conception 15
4.2. Partie II : Prototypage 15

Chapitre 2 : Les systèmes interactifs mixtes 17
1. Définition et terminologie 18
2. Exemples 19
2.1. Des prototypes de recherches en Interaction Homme-Machine 20
2.2. Des jeux vidéo augmentés 24
2.3. Des œuvres d'art 27
2.4. Synthèse des exemples 28
3. Prise de recul 29
3.1. Explosion des possibilités 30
3.2. Liaison entre des mondes différents 33
3.3. Aspects humains 35
4. Synthèse 38
4.1. Motivations 38
4.2. Objectifs 38
4.3. Démarche scientifique 39

Partie I : Conception 41

Chapitre 3 : État de l'art des outils conceptuels 43
1. Tour d'horizon des outils conceptuels 45
1.1. Outils conceptuels pour les interfaces mixtes 45
1.2. Outils conceptuels pour les interfaces tangibles 56
2. Analyse des outils existants au regard des objectifs 65
2.1. Dimensions utiles 66
2.2. Chevauchements 66
2.3. Incomplétude 67
3. Synthèse 69

Chapitre 4 : La solution proposée, le modèle d'interaction mixte 71
1. Modèle d'interaction mixte : centrer l'interaction entre l'utilisateur et le système autour des objets mis en jeu 73
1.1. Motivations 73
1.2. Description d'un objet mixte 73
1.3. Description de l'interaction avec des objets mixtes 77
1.4. Conclusion sur la description de l'interaction centrée sur les objets 83
2. Espace de caractérisation engendré 84
2.1. Caractérisation intrinsèque 84
2.2. Caractérisation extrinsèque 95
2.3. Synthèse des caractéristiques 102
3. Synthèse 107

Chapitre 5 : Carnet de route et évaluation du modèle d'interaction mixte 109

1. Validation d'un modèle d'interaction 110
1.1. Approches existantes 110
1.2. Apports et limites 111
2. Évaluations conceptuelles 112
2.1. Inspection selon les dimensions cognitives des notations 112
2.2. Inspection selon [Olsen, 2007] 113
3. Carnet de route et études expérimentales 114
3.1. Expériences de conception réelles 114
3.2. Évaluation avec protocole auprès des concepteurs 160
4. Synthèse 172

De la modélisation conceptuelle au prototypage 173

Partie II : Prototypage 175

Chapitre 6 : État de l'art des outils de prototypage 177
1. Objectifs 178
1.1. Pour le prototypage isolé 179
1.2. Pour l'aspect prototypage de la conception 180
2. Tour d'horizon des outils de prototypage 180
2.1. Outils matériels 180
2.2. Outils logiciels 185
2.3. Disponibilités 203
3. Analyse des outils existants au regard des objectifs 205
4. Synthèse 207

Chapitre 7 : La solution proposée, la boîte à outils OP 209
5. Description de la boîte à outils OP 210
5.1. Éléments Constitutifs : Composants OP 210
5.2. Structure de la boîte à outils 212
5.3. Outil graphique de mise au point d'un objet mixte 215
6. Composants OP disponibles 217
6.1. Composants pour les dispositifs de liaison 218
6.2. Composants pour les langages de liaison 224
6.3. Composant de composition de modalités de liaison 229
6.4. Composant pour les propriétés numériques 230
7. Exemple d'utilisation : objet sensible à la lumière 230
7.1. Prototypage de l'objet 231
7.2. Interface graphique 232
7.3. Exemples de modifications 233
8. Insérer un objet mixte dans une application 234
8.1. Insérer un objet dans une application développée avec Qt 234
8.2. Insérer un objet dans une application développée sans Qt 234
9. Synthèse 235

Chapitre 8 : Carnet de route et évaluation de la boîte à outils OP 237
1. Évaluation des boîtes à outils 238
1.1. Cadres d'évaluation 238
1.2. Analyse d'évaluations d'outils 242
2. Carnet de route et évaluations de la boîte à outils OP 251
2.1. Évaluation conceptuelle de la boîte à outils OP 251
2.2. Carnet de route et études expérimentales en situation réelle d'utilisation de la boîte à outils OP 254
2.3. Évaluation expérimentale en laboratoire de la boîte à outils OP 265
3. Synthèse 284

Chapitre 9 : Conclusion et Perspectives 287
1. Résumé des contributions 288
1.1. Contribution pour la conception : le modèle d'interaction mixte 288
1.2. Contribution pour le prototypage : la boîte à outils OP 289
2. Limites identifiées et extensions à court terme 289
3. Perspectives à moyen terme 290
3.1. Pour le modèle d'interaction mixte 290
3.2. Pour l'outil de prototypage OP 291
4. Ouverture 292
4.1. Interaction collaborative 292
4.2. Conception par l'utilisateur final 293

Tables des Figures 295

Publications 307

Références 309

Annexe A : Modélisation des exemples 317
1. DigitalDesk (bouton de papier FILL) 318
2. SandScape 319
3. Actuated workbench 320
3.1. Manipulation avec la trackball 320
3.2. Manipulation directe 321
4. PICO 321
5. CASPER, première version 322
6. Carcade 323
7. The Golden Calf 326
8. Alexitimia 326
9. Voice Boxes 327
10. LiveWire 329
11. The Eye Of Judgment 330
12. reacTable 331
13. ARQuake 332
14. Caméléon 334

Annexe B : Solutions conçues par le focus group 335
1. Solutions générées pour l'objet de la tâche 336
2. Solutions générées pour l'interaction 337
2.1. Barre d'onglets 338
2.2. Levier 343
2.3. Outils à poser 345

Annexe C : Questionnaires d'évaluation de la compréhensibilité 347
1. Questionnaire A 348
2. Questionnaire B 365

Annexe D : Spécifications détaillées de OP 383
1. RampInputLanguage 384
2. ThresholdInputLanguage 385
3. DelayLanguage 385
4. RepeatLanguage 386
5. ShortDisplayOutputLanguage 386
6. BeepOutputLanguage 387

Annexe E : Questionnaire d'évaluation de OP 389

Chapitre 1 : Introduction

1. Constat

Nos travaux de recherche s'inscrivent dans le domaine de l'Interaction Homme-Machine (IHM). Le thème de recherche est celui des systèmes de réalité mixte, les systèmes informatiques interactifs qui allient les avantages du monde physique et du monde de l'ordinateur.

Depuis l'ENIAC en 1946, l'évolution des interfaces homme-machine nous a conduit des premières interfaces à cartes perforées, en passant par les terminaux de commandes, à une large adoption aujourd'hui des interfaces graphiques. Les étapes de cette évolution ont progressivement rapproché l'ordinateur de ses utilisateurs. Un exemple décisif de cette évolution est la souris inventée en 1968 par Doug Engelbart. Un autre exemple plus récent est l'utilisation directe des doigts sur l'écran de l'iPhone. Ces deux exemples illustrent l'avantage de combiner le monde physique (ici, la manipulation directe des objets [Schneiderman, 1997]) et l'ordinateur (stockage de l'information et rapidité de calcul). La volonté d'allier les avantages du monde physique et du monde de l'ordinateur a contribué à la démocratisation de l'ordinateur.

Poursuivant naturellement cette évolution des interfaces, les systèmes de réalité mixte sont nés de la volonté de bénéficier des capacités informatiques, sans avoir d'une part à se couper du monde physique, ni d'autre part devoir se priver de nos moyens d'interaction avec le monde physique [Wellner, 199]). En nous référant au Digital Desk, le premier exemple de système de réalité mixte [Wellner, 1991], l'utilisateur dessine avec ses crayons habituels, mais peut aussi effectuer des opérations nouvelles comme le copier/coller ou le redimensionnement de dessins. Nous observons dans l'usage de ce système de réalité mixte la volonté de laisser l'utilisateur dans le monde physique et d'offrir de nouvelles actions possibles grâce à l'ordinateur. Les systèmes de réalité mixte permettent donc de sortir les capacités de l'ordinateur de sa boîte grise habituelle, et l'espace d'interaction ne se limite plus à l'ordinateur de bureau avec un clavier, une souris et un écran. L'espace d'interaction devient alors plus vaste, il comprend l'environnement physique et ne se limite plus seulement à un ordinateur sur un bureau. Ainsi les objets physiques deviennent des supports à l'interaction. Par exemple les Media Blocks [Ishii et al., 1997] reposent sur l'utilisation d'icônes physiques (des cubes) pour manipuler des fichiers informatiques.

Cette volonté de fusion des capacités de traitements informatiques avec l'environnement physique a suscité de nombreux travaux de recherche en Interaction Homme-Machine dans le domaine de la réalité mixte. Il existe néanmoins un fossé entre l'évolution des interfaces des systèmes commerciaux et les systèmes de réalité mixte développés en recherche. Pourtant, l'avantage de la réalité mixte est important : elle permet d'abolir la frontière entre le monde physique et l'ordinateur en rendant possible une interaction fluide et harmonieuse avec des éléments issus du monde physique et du monde numérique (de l'ordinateur). Malgré ces avantages, peu de techniques d'interaction mixte conçues par la recherche sont popularisées. Les raisons de cette fracture résident dans la nécessité pour les concepteurs d'avoir des modèles et des outils pour concevoir de tels systèmes [Beaudouin-Lafon, 2004]. Or les modèles d'interaction et outils existants sont inadaptés et ne permettent pas de généraliser les connaissances des concepteurs aux nouvelles techniques d'interaction mixte.

2. Objectifs

Dans ce contexte, l'objectif de notre travail est de fournir aux concepteurs de systèmes de réalité mixte un cadre pour l'exploration systématique de l'espace de conception. Pour cela, nous visons à fournir des outils dédiés aux activités de conception qu'elles soient conceptuelles ou pratiques (Figure 1).

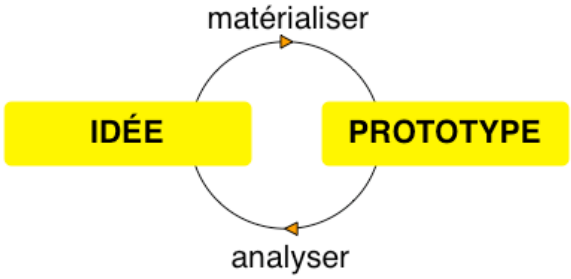


Figure 1 : Deux facettes de la conception : activités conceptuelles (idée) et pratiques (prototype) de la conception.

En effet, « […] *design is a continuous coupling of internal mental activities and external realization activities* » [Lim et al., 2008]. À la Figure 1, nous représentons cette transition entre activité mentale et activité de réalisation de prototype sous la forme de micro-itérations :

1. Un concepteur a une idée.
2. Il la matérialise en construisant un prototype.
3. Ce prototype lui permet de raisonner et de communiquer sur son idée auprès d'autres concepteurs de son équipe.
4. Le(s) concepteur(s) peuvent alors analyser ce prototype en le manipulant pour rebondir sur une nouvelle idée.

Aussi, nous visons l'aide à la conception en accompagnant le concepteur dans ces micro-itérations avec un modèle d'interaction et un outil de prototypage adapté au modèle d'interaction :

- Un modèle d'interaction intervient dans le cycle de la Figure 1 pour aider le concepteur à trouver des idées en explorant de façon systématique l'espace de conception et permet ainsi de structurer l'activité mentale. Il permet également de matérialiser les idées sous la forme de schémas ou d'analyser les prototypes en les modélisant.
- Un outil de prototypage rapide intervient dans le cycle de la Figure 1 pour aider le concepteur à construire des prototypes. Un outil de prototypage adapté au modèle d'interaction permet une transition fluide entre la conception avec le modèle d'interaction et la matérialisation des idées conçues sous forme de prototypes. Il permet donc de matérialiser les idées pour produire un support tangible à l'analyse afin de rebondir vers une nouvelle idée ou un affinement de l'idée initiale.

3. Contributions

Nos contributions concernent les facettes conceptuelles et pratiques de la conception (Figure 2).

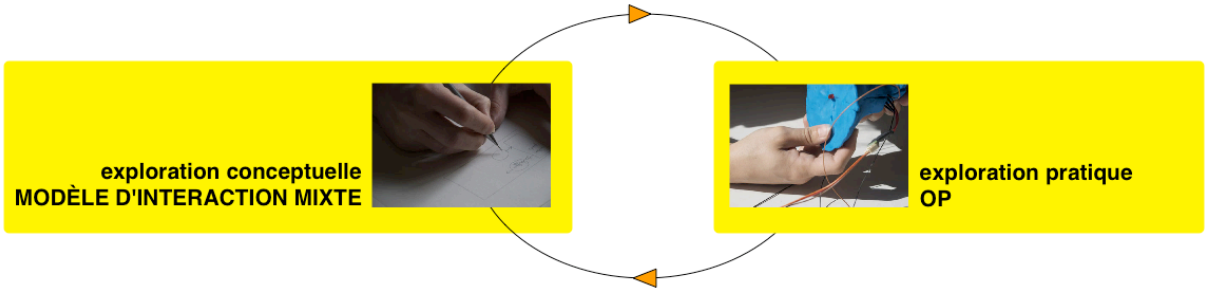


Figure 2 : Contribution conceptuelle et pratique de nos travaux.

3.1. Contribution conceptuelle

Pour comprendre, cerner, et contribuer à l'espace de conception des systèmes de réalité mixte, nous proposons un modèle d'interaction [Appert, 2007][Beaudoin-Lafon, 2004], appelé modèle d'interaction mixte (Figure 2, gauche). Il permet au concepteur de :

- décrire les solutions de conception envisagées,
- comparer plusieurs alternatives de conception,
- explorer l'espace des possibilités.

Le modèle d'interaction mixte unifie les travaux existants concernant la conception des systèmes de réalité mixte, pour obtenir une compréhension globale de ces systèmes. En utilisant le modèle d'interaction mixte, un concepteur doit savoir comment utiliser les travaux conceptuels existants de façon systématique pendant la conception.

3.2. Contribution pratique

Comme souligné dans [Beaudoin-Lafon, 2004], un paradigme d'interaction nécessite à la fois un modèle d'interaction et des outils correspondants pour faciliter le prototypage. Afin de rendre opérationnel notre modèle d'interaction, nous avons donc étudié les outils de prototypage existants à la lumière de notre modèle d'interaction et nous avons développé un outil de prototypage adapté, appelé OP (*Object Prototyping*) (Figure 2, droite).
Les développeurs doivent être capables de produire des prototypes ou ébauches, basés sur les concepts clés de l'espace de conception, tout en profitant des avantages des outils de prototypages existants.

4. Structure du mémoire

La structure du mémoire reflète les deux facettes, conceptuelle et pratique, de nos travaux centrés sur la conception des systèmes de réalité mixte : après un deuxième chapitre qui situe nos travaux dans le contexte de la recherche sur les systèmes de réalité mixte, la thèse est organisée en deux parties : Conception et Prototypage. Nous adoptons une structure identique pour ces deux parties (Figure 3) en trois volets que sont l'existant, notre solution et le carnet de route de son utilisation et son évaluation.

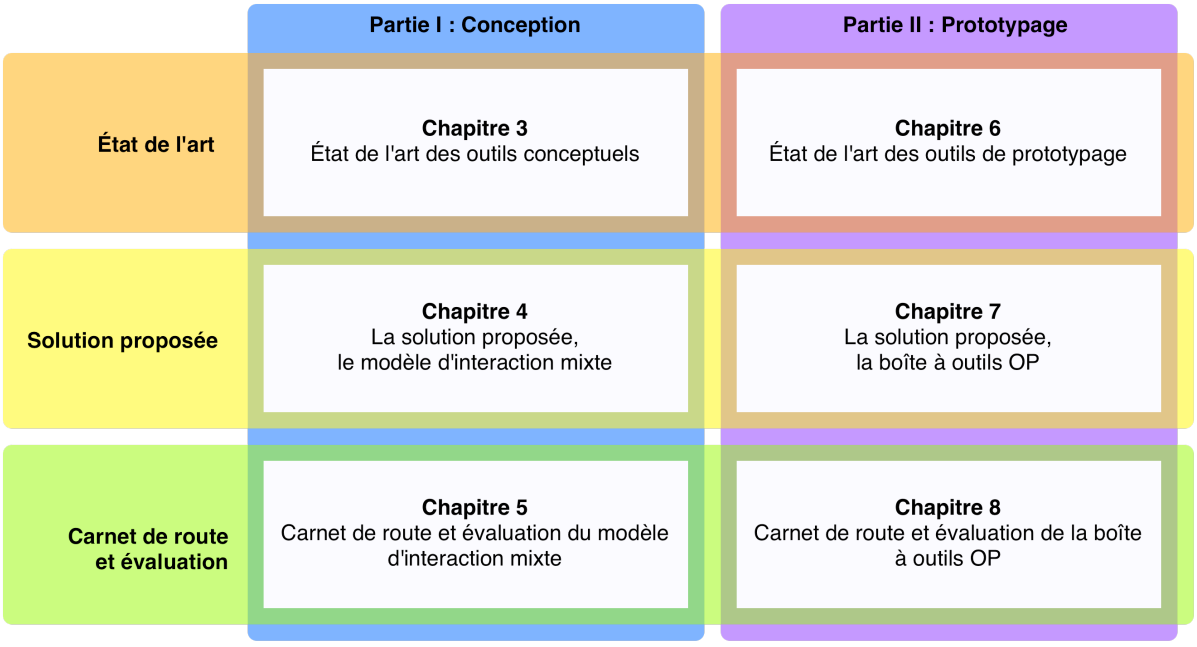


Figure 3 : Structure du mémoire.

En introduction à ces deux parties, nous définissons dans le **Chapitre 2** les systèmes de réalité mixte et nous les illustrons avec des exemples issus des divers domaines d'application ainsi que des communautés qui s'y intéressent. Nous exposons également les questions de recherche rencontrées concernant les interfaces de réalité mixte. Une fois le contexte de nos travaux de recherche précisé,

nous présentons en conclusion les motivations, les objectifs, et la démarche scientifique adoptée pour ces travaux.

4.1. Partie I : Conception

Le **Chapitre 3** dresse d'abord un état de l'art des outils conceptuels pour la conception des interfaces de réalité mixte. Nous présentons ensuite une analyse de la contribution de ces travaux au regard de nos objectifs. Nous montrons quelles sont les dimensions utiles à la conception qu'ils apportent au domaine, ainsi que leurs couvertures, leurs chevauchements et leurs manques.

Le **Chapitre 4** introduit un nouveau modèle d'interaction, le *modèle d'interaction mixte*. Tout d'abord, nous décrirons l'entité centrale à notre modèle, un objet mixte, puis l'interaction mettant en jeu de tels objets mixtes. Nous analysons le pouvoir descriptif du modèle d'interaction mixte grâce aux exemples introduits dans le premier chapitre (Chapitre 2).
Nous étudions ensuite son pouvoir comparatif en considérant des exemples d'interfaces de réalité mixte du Chapitre 2 qui sont proches et dont il est difficile d'appréhender les différences. Pour l'étude du pouvoir comparatif, nous considérons d'abord l'objet mixte indépendamment du contexte applicatif (caractéristiques intrinsèques). Nous considérons ensuite l'objet mixte dans son contexte applicatif (caractéristiques extrinsèques).

Le **Chapitre 5** est consacré à l'apport du modèle du point de vue de son utilisation en conception, c'est-à-dire l'évaluation du pouvoir génératif du modèle d'interaction mixte. Nous abordons tout d'abord les problèmes liés à la validation d'un tel outil conceptuel, et nous présentons ensuite le carnet de route de l'utilisation du modèle d'interaction mixte ainsi que les évaluations que nous avons menées, autant théoriques qu'expérimentales.

4.2. Partie II : Prototypage

Le **Chapitre 6** présente un état de l'art des outils de prototypage pour les systèmes de réalité mixte. Nous analysons ces outils au regard de nos propres objectifs de prototypage des interfaces mélangeant éléments physiques et numériques, conçues avec le modèle d'interaction mixte.

Le **Chapitre 7** introduit une boîte à outils appelée OP (*Object Prototyping*) qui, tout en s'appuyant sur les outils existants étudiés au Chapitre 6, répond à notre objectif. Dans cette boîte à outils, les éléments à la disposition du développeur correspondent aux éléments du modèle d'interaction mixte. Nous décrivons d'abord la boîte à outils dans son ensemble. Nous présentons ensuite les éléments constitutifs d'OP qui sont disponibles aujourd'hui. Nous montrons ensuite sur un exemple comment utiliser OP pour prototyper un objet mixte. Enfin, nous présentons comment insérer dans une application un objet mixte prototypé avec la boîte à outils OP.

La **Chapitre 8** est consacré à l'évaluation de la boîte à outils OP : ses apports et ses limitations. Nous présentons d'abord les enjeux et les problématiques liés à l'évaluation des outils pour l'interaction homme-machine. En effet, il est nécessaire de remettre l'évaluation de notre boîte à outils OP dans son contexte, car la difficulté de l'évaluation des outils pour l'interaction homme-machine est reconnue, et les formes d'évaluation proposées dans la littérature ont influencé la façon dont l'évaluation de notre boîte à outils a été menée. Après avoir dressé un état de l'art des méthodes d'évaluation, nous présentons ensuite le carnet de route de l'utilisation de la boîte à outils OP et les différentes formes d'évaluation que nous avons menées.

Le **Chapitre 9** conclut nos travaux en présentant un résumé des contributions. Nous ouvrons ensuite sur des perspectives à moyen et long terme.

Enfin, quatre annexes sont fournies en fin de mémoire. L'**Annexe A** présente l'ensemble des exemples utilisés dans la Partie I modélisés selon le modèle d'interaction mixte. L'**Annexe B** présente les solutions conçues lors de l'évaluation du modèle d'interaction mixte en groupe focalisé. L'**Annexe C** présente les questionnaires fournis aux participants aux évaluations de la compréhensibilité du modèle d'interaction mixte. L'**Annexe D** présente en détail les spécifications de la boîte à outils OP. L'**Annexe E** inclut les questionnaires fournis aux participants des évaluations de la boîte à outils OP.

Chapitre 2 : Les systèmes interactifs mixtes

Le cadre de ce travail doctoral est l'Interaction Homme-Machine, et plus précisément la réalité mixte. Les systèmes interactifs de réalité mixte suscitent un intérêt grandissant parmi de plus en plus de concepteurs, bien qu'ils ne soient pas autant répandus parmi le grand public que les interfaces graphiques. Nous présentons dans ce chapitre les définitions et les enjeux des systèmes de réalité mixte, et nous situons ces travaux dans leur contexte.
Nous posons d'abord les bases de la réalité mixte en présentant une définition de ce paradigme d'interaction et la terminologie utilisée. Nous illustrons ensuite les interfaces de réalité mixte avec des exemples issus des divers domaines d'application ainsi que des communautés qui s'y intéressent. Enfin, nous exposerons les questions de recherches rencontrées dans la littérature et qui concernent ces systèmes. Une fois le contexte des travaux de recherches précisé, nous présentons en conclusion les motivations, les objectifs, et la démarche scientifique adoptée pour ce travail.

1. Définition et terminologie

L'informatique, et le monde numérique qu'il engendre, offre des capacités de stockage et de traitement de l'information. Pour profiter de ces capacités dans le monde physique dans lequel nous vivons, nous devons utiliser des interfaces. Les systèmes de réalité mixte ont la spécificité d'inscrire dans l'environnement physique les interfaces entre monde physique et monde numérique, au contraire des interfaces graphiques (Figure 4) et de la réalité virtuelle (Figure 5). Même si toute interface offre par définition un lien avec le monde physique, les interfaces graphiques offrent un lien réduit à la plateforme standard clavier, souris et écran. La réalité virtuelle quant à elle cherche à isoler l'utilisateur du monde physique pour l'immerger dans un monde généré par le système informatique.

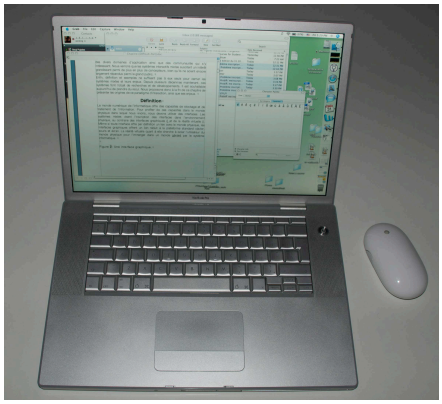


Figure 4: Une interface graphique, avec clavier, écran et souris.

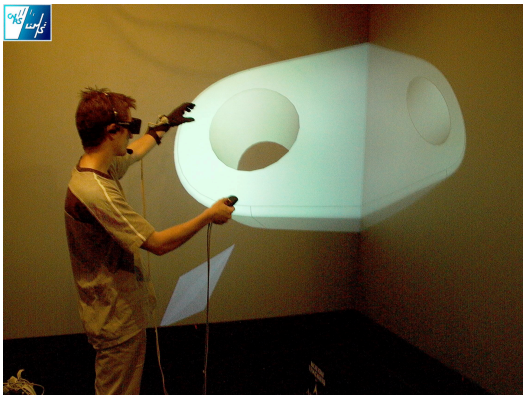


Figure 5: Une interface de réalité virtuelle. Illustration extraite de http://www.limsi.fr/IMAGES2/thomas.jpg.

Dans le cas des systèmes de réalité mixte, les études définissent les liens entre le monde physique et le monde numérique [Renevier, 2004], ou encore l'augmentation du physique par le numérique ou inversement [Dubois, 2001]. Une augmentation ou un lien désigne le processus par lequel est associé une propriété physique avec une propriété numérique. Par exemple dans [Carvey et al., 2006] le poids d'un objet (propriété physique) est associé à un fichier (propriété numérique). Le lien entre les deux est effectué par une balance USB (Figure 6, gauche, en blanc).



Figure 6: Un lien ou augmentation mis(e) en place grâce au poids et à une balance USB. Illustration extraite de [Carvey et al., 2006].

La terminologie concernant les systèmes de réalité mixte est encore soumise à évolution. Nous trouvons très couramment les termes de réalité augmentée, réalité mixte, virtualité augmentée, ou interfaces tangibles.

- Le terme de réalité augmentée désigne les interfaces qui fournissent de l'aide grâce à l'informatique à une activité que l'utilisateur effectue dans le monde physique. La chirurgie assistée par ordinateur [Dubois, 2001] ou encore le tourisme [Feiner et al., 1997] sont des applications dont les interfaces ont été les premières à s'inscrire dans cette définition de la réalité augmentée.
- La virtualité augmentée est un terme utilisé dans [Dubois, 2001] par exemple, pour désigner les interfaces qui permettent de réaliser dans monde physique une tâche qui s'effectue dans le monde numérique. C'est le cas par exemple de la gomme qui est utilisée pour effacer du texte projeté sur la feuille dans le DigitalDesk [Wellner, 1993].
- La réalité mixte est un terme plus général qui regroupe les deux précédentes classes d'interfaces.
- Les systèmes mixtes désignent les systèmes de réalité mixte. Le terme de systèmes est un abus de langage, car ce n'est pas le système qui est mixte mais l'interface avec le système (numérique).
- Les interfaces tangibles sont des interfaces où les objets mis en jeu dans l'interaction sont tangibles, palpables.

Les interfaces mixtes regroupent donc de nombreux types d'interfaces, puisque certains de ces termes, comme les interfaces tangibles, désignent à eux seuls un champ de recherche à part entière, avec leurs propres conférences, comme *Tangible and Embedded Interaction*[1] pour les interfaces tangibles.
Les interfaces de réalité mixte définissent un vaste axe de recherche que nous considérons à la fois comme intégrateur de plusieurs paradigmes, mais aussi orthogonal à d'autres axes comme les collecticiels et la mobilité. Nous détaillons plus en avant le positionnement des systèmes de réalité mixte à la section 3 de ce chapitre.

Nous venons de présenter la définition des interfaces mixtes. Afin de l'illustrer, nous présentons maintenant des exemples d'interfaces mixtes.

2. Exemples

Afin d'illustrer la diversité du domaine, le choix des exemples de systèmes que nous présentons maintenant repose sur la volonté d'illustrer des points particuliers et variés de l'espace de conception : interfaces anciennes ou récentes, interfaces tangibles ou intangibles, interfaces de réalité augmentée ou de virtualité augmentée, interfaces mobiles ou non, interfaces collaboratives ou non. De plus, nous soulignons également la diversité des disciplines, de la recherche en Interaction Homme-Machine aux arts plastiques, en passant par le jeu vidéo et le design.

[1] Interaction tangible et embarquée

2.1. Des prototypes de recherches en Interaction Homme-Machine

2.1.1. Des surfaces interactives

2.1.1.1. Le DigitalDesk

En Interaction Homme-Machine se trouve un des systèmes les plus anciens qui puisse être qualifié de mixte : le Digital Desk [Wellner, 1993]. Ce système propose plusieurs scénarios liés à l'activité d'un utilisateur sur son bureau, comme la comptabilité, l'écriture, ou le dessin. Nous considérons pour cet exemple la simulation du scénario de dessin. L'utilisateur dessine avec un crayon sur une feuille de papier placée sur une table équipée d'une caméra et d'un projecteur. À la Figure 7, l'utilisateur commence à dessiner des tuiles sur le toit de la maison. Il décide ensuite d'utiliser un bouton en papier où il est écrit *FILL*[2] en le faisant pointer sur les tuiles du toit. Il appuie sur le bouton de papier, et cette action est capturée par la caméra. Le toit est alors rempli de tuiles, qui sont affichées sur la table grâce au projecteur. Le dessin résultant est donc composé de tuiles dessinées au crayon et de tuiles affichées grâce à un projecteur. À la Figure 8, l'utilisateur gomme les tuiles qui ne sont pas dessinées au crayon avec une gomme classique grâce à la caméra située au-dessus de la table : les tuiles gommées ne sont plus projetées sur la feuille.



Figure 7: Remplir le toit de tuiles avec le DigitalDesk.

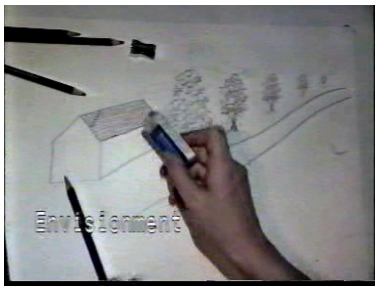


Figure 8 : Gommer une partie des tuiles avec le DigitalDesk.

2.1.1.2. SandScape

Le système SandScape [Ishii et al., 2004] propose une interface tangible utilisant du sable pour comprendre le relief à travers plusieurs simulations (altitude, ensoleillement, écoulement, etc.). Les utilisateurs peuvent voir ces simulations projetées en coïncidence sur le sable qui représente le relief. Les utilisateurs manipulent le sable (Figure 9), et en voient les conséquences en temps réel grâce à la projection. Ils peuvent également changer de visualisation en déplaçant un marqueur tangible d'une niche à une autre : À la Figure 10, l'utilisateur déplace un bloc (le marqueur) et le pose dans une des six niches qui bordent la table. La Figure 11 montre la simulation de l'ensoleillement et la Figure 12 montre une visualisation de l'altitude.

[2] Remplir



Figure 9 : Des utilisateurs interagissant avec le relief en agissant sur le sable. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.



Figure 10 : Un utilisateur change de simulation en déplaçant un marqueur tangible d'un espace à un autre. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.

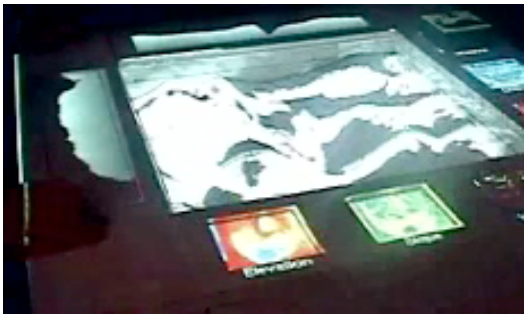


Figure 11: Une visualisation de l'ensoleillement avec SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.

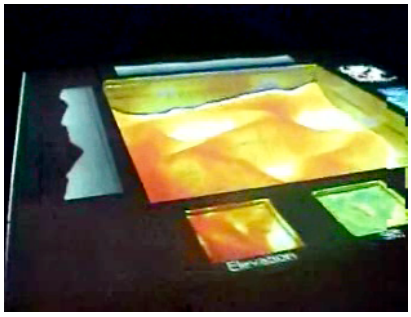


Figure 12: Une visualisation de l'altitude avec SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.

La Figure 13 présente l'installation matérielle du système SandScape ainsi que les éléments participant à l'interaction. Une caméra infrarouge capte l'intensité de l'émission infrarouge émanant de la partie inférieure de la table. L'intensité captée dépend de l'épaisseur du sable que les utilisateurs sont en train de manipuler. Le projecteur affiche alors sur le sable la simulation choisie par le marqueur tangible, tandis qu'une vue en trois dimensions est projetée sur le mur du fond.

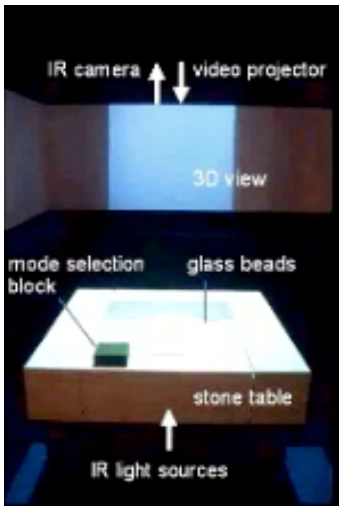


Figure 13 : L'installation matérielle du système SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.

2.1.1.3. PICO

Plus récemment, PICO [Patten, Ishii, 2007], dont le sigle signifie *Physical Intervention in Computational Optimization*[3], est une surface interactive où sont embarqués des aimants. Sur cette table, l'utilisateur ou le système peuvent indifféremment déplacer les objets. Une caméra et un projecteur situés au-dessus de la table permettent respectivement de capter la position des objets et de projeter un retour d'information. Le système calcule la position idéale des objets et la table aimantée les déplace vers cette position (Figure 14). À cette interaction s'ajoute l'utilisation possible de contrainte physique, comme dans la Figure 15 : l'objet ne peut pas accéder à la partie centrale de la table.

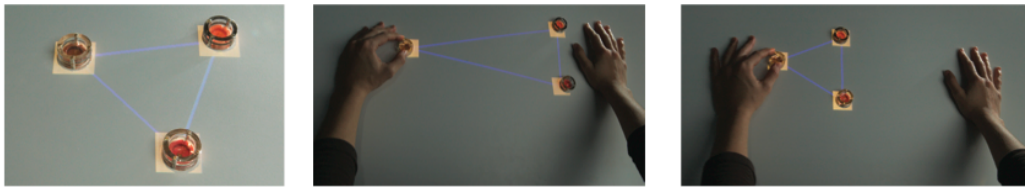


Figure 14 : Le système PICO. Illustration extraite de http://www.jamespatten.com/pico/picovid.html.

[3] Intervention physique dans l'optimisation par ordinateur

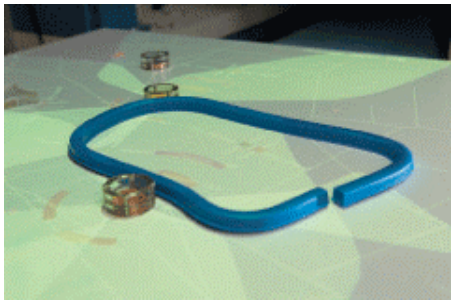


Figure 15 : PICO utilisé avec une contrainte physique. Illustration extraite de [Patten, Ishii, 2007].

Nous venons de présenter l'exemple de trois systèmes de réalité mixte qui prennent la forme de surface interactive. Nous présentons maintenant des exemples de systèmes de réalité augmentée, toujours issus de la recherche en Interaction Homme-Machine, qui s'adressent à des métiers critiques.

2.1.2. Des interfaces de réalité augmentée, pour des métiers critiques

2.1.2.1. Pour les contrôleurs aériens

Pour assister l'activité critique des contrôleurs aériens, le système Caméléon [Mackay, 1996] ne vise pas à remplacer les bandes de papiers utilisés par les contrôleurs aériens. Ces bandes de papiers représentant les avions en vol sont annotés et disposés de façon pertinente par les contrôleurs aériens. Les tentatives de remplacement par un affichage sur un écran avaient toujours été un échec. Après avoir décelé l'importance de ces objets dans le travail des contrôleurs aériens, les concepteurs se sont donc concentrés sur l'augmentation de ces bandes de papiers, de telle façon que le système puisse être utilisé avec ou sans aide informatique. Ainsi, par exemple, avec l'aide du système et des bandes de papier augmentées, les contrôleurs peuvent par exemple souligner un numéro de vol pour visualiser l'avion sur le radar, mais ils peuvent aussi chercher l'avion sur le radar comme ils en ont l'habitude.

2.1.2.2. Pour les chirurgiens

Le système CASPER [Dubois, 2001] (*Computer ASisted PERicardial puncture*[4]) permet l'acquisition et la modélisation préopératoires de la région d'une effusion péricardique à partir de laquelle une trajectoire idéale de l'aiguille de ponction est planifiée. Pendant l'opération, le chirurgien est guidé grâce à un localisateur optique qui capture la position de l'aiguille en trois dimensions. La Figure 16 montre l'application pendant l'intervention (étape de guidage). Sur l'écran, la position et l'orientation de l'aiguille sont représentées par deux croix tandis qu'une croix fixe représente la trajectoire idéale (Figure 17). Quand les trois croix sont superposées, la trajectoire de l'aiguille correspond à la trajectoire idéale.

[4] Ponction péricardiale assistée par ordinateur



Figure 16 : Le système CASPER, première version (d'après [Dubois, 2001]).

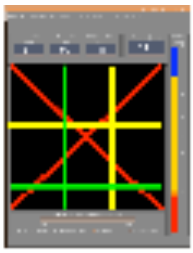


Figure 17 : Le système CASPER, première version, et la visualisation des trajectoires de l'aiguille de ponction sur l'écran (d'après [Dubois, 2001]).

Nous avons présenté des exemples de systèmes issus de la recherche en interaction homme machine. Nous avons pris l'exemple de surfaces interactives ainsi que de systèmes de réalité augmentée pour des métiers critiques. Nous présentons maintenant des systèmes issus du jeu vidéo.

2.2. Des jeux vidéo augmentés

Les jeux vidéos augmentés que nous présentons ont été développés par la recherche ou l'industrie du jeu vidéo.

2.2.1. Par la recherche

Le premier de ces exemples issus de la recherche, Carcade, a été développé par des étudiants en école d'art, alors que le second ARQuake, par des chercheurs en informatique.

2.2.1.1. Carcade

Carcade [Fischer, Luge, Polk, 2008] est la contraction de *car*[5] et *arcade*[6]. Carcade est un jeu pour les voyages en voiture, avec une caméra placée face à la fenêtre de la voiture qui traque le mobilier urbain et le paysage qui défile. Les éléments du paysage ainsi que la façon de conduire deviennent alors une partie intégrante du jeu. L'ordinateur est posé sur les genoux de l'utilisateur. La Figure 18 montre ce qui s'affiche sur l'écran du joueur. À la Figure 18, son avion rencontre un immeuble : Puisque la vue est bloquée par un immeuble, le joueur doit passer par la fenêtre indiquée s'il ne veut pas s'écraser sur l'immeuble et perdre.

[5] voiture

[6] utilisé pour désigner les jeux vidéos, ou salle de jeux électroniques



Figure 18: Carcade : vue de l'écran de l'ordinateur posé sur les genoux de l'utilisateur. Une caméra capture le paysage qui défile par la fenêtre et que l'utilisateur voit sur l'écran. Sur cette capture d'écran, le joueur doit passer par la fenêtre indiquée s'il ne veut pas s'écraser sur l'immeuble.

2.2.1.2. ARQuake

L'ARQuake [Thomas et al., 2000] est une version en réalité augmentée du célèbre jeu Quake. Le joueur explore le monde physique afin de jouer à un jeu généré par le système. Le jeu utilise des récepteurs GPS, un capteur d'orientation, un dispositif d'interaction en forme d'arme à feu, et un ordinateur portable que le joueur porte dans un sac à dos. La Figure 19 présente une vue affichée dans le casque semi-transparent du joueur.



Figure 19 : Une vue dans le casque semi-transparent d'un joueur de l'ARQuake : on y voit à la fois le monde physique et les éléments ajoutés par l'ordinateur.

2.2.2. Par l'industrie du jeu vidéo

Nous présentons deux exemples de jeux en réalité augmentée issus de l'industrie du jeu vidéo : The eye of judgment et Tuttuki Bako.

2.2.2.1. The eye of Judgment

The Eye of Judgment[7] est un jeu pour la console de jeu PS3 de Sony, disponible en octobre 2007 en France. C'est à la fois un jeu de plateau et de console. Les joueurs manipulent des cartes dont l'image est augmentée à l'écran grâce à une caméra. La Figure 20 montre des scènes tirées de la campagne promotionnelle du jeu.

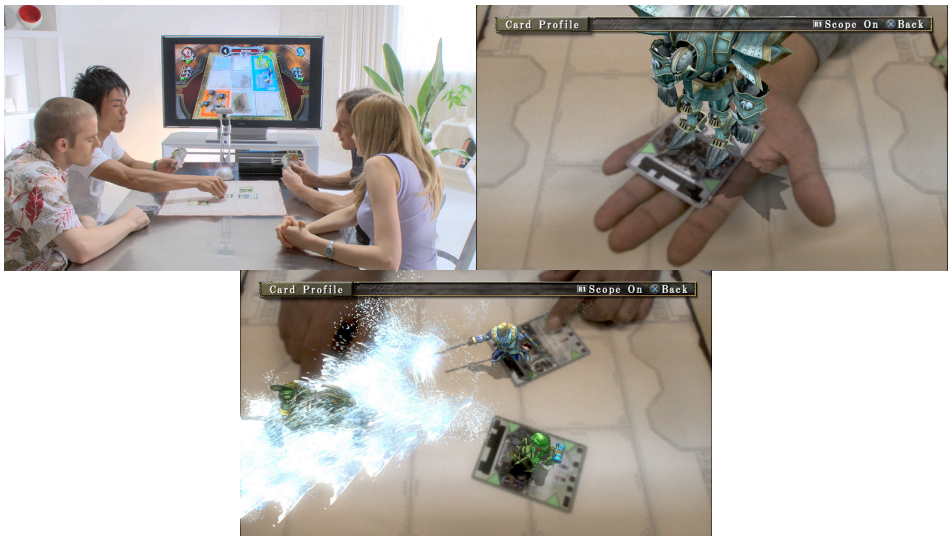


Figure 20: The Eye of Judgment pour PS3 (**d'après** http://www.eyeofjudgment.com/).

2.2.2.2. Tuttuki Bako

Lancé par Bandai au japon, *Tuttuki bako*[8] (http://www.asovision.com/tuttuki/) est un jeu simple doté d'un écran à basse définition où l'utilisateur interagit avec les objets affichés en introduisant et en bougeant son doigt à l'intérieur du boîtier. Une image pixellisée du doigt, superposée, apparaît alors à l'écran. La Figure 21 présente l'interaction avec un objet du jeu : un panda suspendu.



Figure 21: Le jeu Tuttuki Bako (**d'après** http://www.asovision.com/tuttuki/).

[7] L'œil du jugement
[8] Boîte Tuttuki

2.3. Des œuvres d'art

2.3.1. *The Golden Calf*

The Golden Calf[9] [Shaw, 1994] est un système de réalité mixte conçu en 1994 par Jeffrey Shaw, un artiste pionnier dans l'utilisation de l'informatique et de l'interaction dans l'art. Cette œuvre est constituée d'un socle blanc où est posé un moniteur LCD couleur, connecté par un câble à un ordinateur situé dans le socle. Le destinataire de l'œuvre prend cet écran dans les mains. Sur l'écran, il voit une image 3D du socle avec un veau d'or exposé dessus. En déplaçant l'écran autour du socle, l'utilisateur peut regarder le veau d'or depuis différents points de vue. Le comportement des destinataires qui regardent avec fascination au travers de l'écran fait alors écho avec humour l'épisode biblique d'adoration du veau d'or.



Figure 22: *The Golden Calf*, Jeffrey Shaw, 1994 (d'après [Shaw, 1994]).

2.3.2. *LiveWire*

LiveWire est une installation (mixte) conçue par Natalie Jeremijenko en 1995 [Jeremijenko, 1995]. Les mouvements d'un fil suspendu au plafond sont liés au nombre de paquets qui transitent sur le réseau. Plus il y a de trafic sur le réseau local, plus le fil bouge.



Figure 23: *LiveWire*, Natalie Jeremijenko, 1995 (d'après [Jeremijenko, 1995]).

[9] Le veau d'or

2.3.3. *Voice Boxes*

Une *Voice Box* (1995, du même auteur, [Jeremijenko, 1995]) est une boîte cubique de deux pouces de côté qu'on ouvre pour enregistrer du son à l'intérieur, qu'on touche pour en écouter le contenu et qu'on peut déplacer n'importe où, à la manière de post-it vocal et sonore.



Figure 24: *Voice Boxes*, Natalie Jaremijenko, 1995 (d'après [Jeremijenko, 1995]).

2.3.4. *Alexitimia*

Alexitimia est une œuvre de l'artiste Paula Gaetano Adi [Gaetano Adi, 2006]. L'objet ressemble à une grosse goutte qui semble comme du caoutchouc au toucher et qui transpire quand l'utilisateur touche sa surface. Des capteurs de toucher permettent de déclencher la diffusion d'eau venant d'un réservoir caché dans le socle de l'objet.





Figure 25: *Alexitimia*, Paula Gaetano, 2006 (d'après [Gaetano Adi, 2006]).

2.4. Synthèse des exemples

Les exemples considérés soulignent le large éventail de systèmes de réalité mixte existants. Outre le fait que ces systèmes tirent leurs origines de domaines différents, comme la recherche en Interaction Homme-Machine (le Digital Desk, SandScape, PICO, CASPER, Caméléon, ARQuake), l'industrie du jeu vidéo (The Eye of Judgment, ARQuake car tiré de Quake, Tuttuki Bako) et les arts plastiques (Carcade, The Golden Calf, LiveWire, Voice Boxes, Alexitimia), ils sont aussi différents selon d'autres critères :

1. Certaines de ces interfaces datent d'une vingtaine d'années, voire presque 30 ans (The Digital Desk (1993), The golden Calf (1994), LiveWire ou Voice Boxes (1995)), alors que d'autres sont très récentes (The Eye of Judgment (2007), Tuttuki (2008)).
2. Certaines de ces interfaces sont dites tangibles (Alexitimia, Voice Boxes, LiveWire, PICO, SandScape, Caméléon), d'autres non tangibles (The golden Calf, Tuttuki, Carcade, ARQuake). Mais la classification peut être difficile, par exemple pour the Eye of Judgment, le Digital Desk, ou CASPER.
3. Certaines augmentent le monde physique en lui ajoutant des éléments numériques (CASPER, Caméléon, Carcade, The Eye of Judgement), d'autres augmentent le monde numérique en lui ajoutant des éléments physiques (LiveWire, The Golden Calf). D'autres semblent répondre à ces deux critères, comme le Digital Desk, PICO ou SandScape.

4. Certaines de ces interfaces sont conçues spécialement pour un utilisateur mobile : Dans Carcade l'utilisateur est mobile car dans une voiture mobile, dans ARQuake l'utilisateur se déplace, avec The Golden Calf l'utilisateur est incité à tourner autour du socle. D'autres interfaces ne tiennent pas compte de la mobilité et peuvent être utilisée dans tous les cas (Tuttuki Bako, Voice Boxes). Encore d'autres interfaces ne permettent pas la mobilité (Alexitimia, LiveWire, le Digital Desk, PICO, SandScape).
5. Nous distinguons aussi des interfaces conçues pour l'utilisation par plusieurs utilisateurs, dites collaboratives, comme Caméléon, ARQuake, The Eye of Judgment, d'autres qui ne le permettent pas (le Digital Desk, CASPER, Carcade, The golden Calf, Tuttuki) ou encore celles qui sont indifférentes au nombre d'utilisateurs (SandScape, PICO, Voice Boxes, LiveWire, Alexitimia).

Ce tour d'horizon rapide permet d'illustrer la diversité de la réalité mixte. Il nous permet aussi de souligner que les frontières de cet ensemble d'interfaces sont floues, et souvent soumises à débat. Dans nos travaux, nous nous intéressons à examiner les particularités et problématiques que ces interfaces partagent, tout en prenant en compte leur diversité.
Il est important de souligner également que pour très peu de ces systèmes (mais Caméléon en est un exemple), les auteurs reportent les alternatives de conception envisagées. Ceci montre que les systèmes sont conçus de façon ad hoc et illustre le manque de maturité du domaine.

Nous avons présenté des exemples de systèmes de réalité mixte dans cette deuxième section du Chapitre 2, après en avoir donné une définition en première section. Nous proposons maintenant de prendre du recul sur ces interfaces, en faisant un état de l'art des axes de recherches qui les concernent.

3. Prise de recul

Dans [Grudin, 1990], l'origine des interfaces mixtes est expliquée en proposant l'idée que les interfaces ont commencé par être centrées sur la machine pour de plus en plus se tourner vers le monde de ses utilisateurs, c'est-à-dire son monde physique et social. [Dourish, 2004] reprend cette perspective en expliquant l'émergence de la réalité augmentée par l'exploitation nouvelle de capacités liées à notre expérience du monde physique, même si certaines particularités des interfaces graphiques préfiguraient déjà cette tendance, comme la manipulation directe [Schneiderman, 1983]. En d'autres termes, l'émergence de ces interfaces mixtes peut s'expliquer par la volonté de réduire le fossé entre les mondes physique et numérique, décrit dans la théorie de l'action [Norman, 1986]. Selon cette théorie, l'utilisateur est confronté à un gouffre entre son monde et celui du système informatique (Figure 26). D'un côté se trouve l'intention de l'utilisateur. De l'autre se trouve le système informatique. Pour agir sur le système, l'utilisateur devra traduire son intention en actions, faisables avec les dispositifs physiques présents en entrée de la machine. En retour, le système renvoie des signaux sur ses dispositifs physiques en sortie, que l'utilisateur doit interpréter et évaluer.

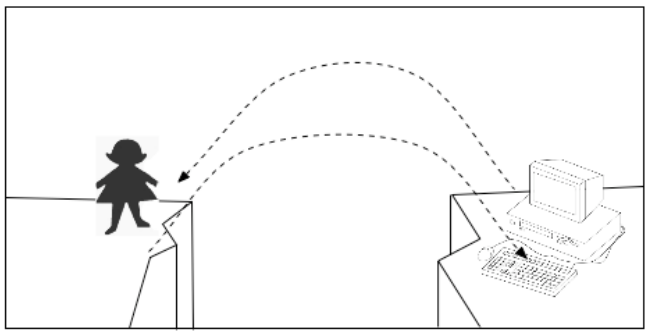


Figure 26 : Théorie de l'action [Norman, 1986].

Cette théorie montre l'indirection entre l'utilisateur et le système : Il est nécessaire pour l'utilisateur d'adapter son intention au système, mais aussi d'évaluer et interpréter sa réponse. Les interfaces de

type terminaux de commande (type clavier/écran) ou graphique (type écran/souris) offrent à l'utilisateur des ressources contraignantes pour traduire ses intentions. Les premières le forcent à traduire ses intentions sous la forme d'un langage limité, qui est loin de la façon dont il agit dans le monde physique. Les secondes s'en approchent, mais n'offrent toujours pas la richesse des interactions du monde physique. C'est à partir de ce constat qu'ont émergé les systèmes de réalité mixte comme la réalité augmentée ou les interfaces tangibles.

Même si ce fossé entre mondes physique et numérique est un frein pour l'interaction avec les systèmes informatiques, et que la réduction de ce fossé fut l'avantage principalement avancé en faveur des interfaces mixtes au moment de leur émergence, il convient d'aller au-delà de cette théorie. Dans cette optique, ce chapitre propose de prendre du recul et de remettre en contexte les interfaces mixtes, en identifiant plus précisément quels sont les aspects qui ont motivé et entretenu la recherche dans ce domaine depuis sa création. À partir de la littérature, nous avons identifié les axes suivants : l'explosion des possibilités, la liaison entre des mondes différents, et les aspects humains.

3.1. Explosion des possibilités

Dans de nombreux travaux sur les interfaces mixtes, nous avons trouvé des problématiques liées à l'explosion des possibilités en matière de conception.
Les concepteurs se sont trouvés face à des alternatives au trio clavier/écran/souris. Les possibilités de couplage entre physique et numérique se sont multipliées, avec de nouveaux objets, contenant d'autres capteurs et d'autres effecteurs. Ceci doit être pris en compte pendant la conception.
L'explosion des possibilités de couplage entre physique et numérique a des conséquences sur le prototypage. Les méthodes de construction de prototypes en papier par exemple, adapté à l'écran des interfaces graphiques, s'appliquent moins bien pour les systèmes de réalité mixte.
L'explosion des possibilités concerne aussi l'utilisation de l'espace. Avec ou sans fil, les ressources d'interaction peuvent être distribués dans l'espace.
Comme nous allons le voir, chacun de ces trois points liés à l'explosion des possibilités de conception sont traités dans la littérature. Nous les détaillons maintenant l'un après l'autre.

3.1.1. Multiplication des possibilités de couplage entre physique et numérique

Nous sommes physiques et nous interagissons avec des objets du monde physique. Au lieu d'avoir une fenêtre unique sur le monde numérique, avec des outils standards tels que le clavier, la souris, l'écran, voire les haut-parleurs, les interfaces mixtes augmentent les possibilités d'associations entre éléments physiques et éléments numériques. Par exemple dans Caméléon [Mackay, 1998], les bandes de papier utilisées par les contrôleurs aériens ont été augmentées de plusieurs façons : des tablettes graphiques utilisées avec un stylo, des écrans tactiles, des caméras, des projecteurs, etc. Pour qualifier cette multiplication des possibilités illustrée sur l'exemple de [Mackay, 1998], nous trouvons dans la littérature l'expression d'augmentation de la bande passante entre utilisateur et système.
Ainsi, nous pensons que les interfaces graphiques forment un sous-ensemble restreint des interfaces mixtes. Ce type d'interface a fait l'objet depuis des décennies de recherches approfondies et a répandu une forme de standard surtout via sa commercialisation à grande échelle. En effet il est difficile de dire que la souris n'est pas un objet physique qui est lié au monde numérique, et pourtant un consensus existe sur la différence entre les interfaces mixtes et graphiques. La recherche a donc besoin de distinguer ce sous-ensemble pour signifier qu'elle s'intéresse à un problème plus large, qui va au-delà de cette plateforme standard d'interaction. La multiplication des possibilités de couplages non-standards entre physique et numérique impose donc la rupture avec les interfaces graphiques.

Les possibilités se multiplient à plusieurs niveaux : elles concernent les objets physiques mis en jeu lors de l'interaction, mais aussi les possibilités technologiques comme les dispositifs de capture et de matérialisation.
Ces liens de nouvelle nature entre le physique et le numérique peuvent se faire via des objets autres que la souris, le clavier ou l'écran. Par exemple dans [Ishii, Ullmer, 1997] sont évoqués les objets quotidiens, les bâtiments. Les objets du monde physique servent alors d'interfaces avec le monde numérique. Il s'agit donc de réduire le fossé entre les mondes physiques et numériques via les objets physiques pouvant entrer en jeu dans l'interaction avec les systèmes informatiques.
Il y a aussi réduction du fossé entre les mondes physiques et numériques via la multiplication des dispositifs de capture et de matérialisation du système et des algorithmes permettant au système d'obtenir de l'information dans des données brutes capturées. Les actions possibles pour interagir avec le système sont plus nombreuses. Ceci réduit le fossé entre les mondes physiques et numériques. Cette idée est développée dans [Benford et al., 2005], où les auteurs proposent d'étudier les actions attendues de la part des utilisateurs, en regard des actions qui peuvent être captées et de l'information nécessaire pour le système.

Nous présentons maintenant comment l'explosion des possibilités a des conséquences sur le prototypage.

3.1.2. Difficultés pour prototyper

Parmi les enjeux qui découlent de l'apparition des interfaces mixtes, nous trouvons des enjeux qui sont liés au prototypage. La multiplication des possibilités de couplages entre physique et numérique implique de nouvelles problématiques pour le prototypage des interfaces mixtes [Myers et al., 2000, partie 3.2.2]. Les possibilités deviennent nombreuses quant aux matériaux physiques à utiliser, mais aussi la partie matérielle, et par conséquent la partie logicielle du système qui ne peut plus reposer sur les boîtes à outils graphiques.
Nous avons identifié les enjeux suivants :

- La difficulté de prototyper une technique d'interaction nécessitant une expertise particulière, pour un non-informaticien [Hartmann et al., 2007] [Hartmann et al., 2006], voire même pour un informaticien [Greenberg, Fitchett, 2001] [Klemmer et al., 2004],
- Le besoin d'intégrer prototypage physique et logiciel, l'un et l'autre étant interdépendant [Hudson, Mankoff, 2006],
- La nécessité d'avoir des outils de prototypages adaptés aux modèles conceptuels émergents [Beaudoin-Lafon, 2004] pour les interfaces mixtes,
- Le besoin d'avoir des boîtes à outils logicielles facilement extensibles [Myers et al., 2000, partie 3.6. 3ème paragraphe], puisque les systèmes de réalité mixte multiplient les possibilités d'interaction.

Outre l'implication sur les difficultés de prototypage, l'exposition des possibilités concerne aussi l'utilisation de l'espace. C'est ce que nous présentons maintenant.

3.1.3. Utilisation de l'espace

À la lecture de la littérature, nous constatons que l'explosion des possibilités concerne aussi l'espace. Nous présentons tout d'abord la diffusion des systèmes dans l'espace, puis les liens entre le multiplexage et les interfaces mixtes.

3.1.3.1. Diffusion dans l'espace

Les systèmes de réalité mixte sont distribués dans l'espace. En effet, de nombreux travaux initiateurs d'interfaces mixtes proposent de diffuser l'informatique dans l'espace. Nous trouvons par exemple l'intention de « situer l'interaction dans les objets quotidiens et les bâtiments » [Ishii, Ullmer, 1997]. Un exemple concret peut être trouvé dans l'ARQuake, où les utilisateurs doivent utiliser l'espace pour interagir. Un autre exemple se trouve dans SandScape où l'interaction est distribuée

spatialement entre le sable, les niches où poser le marqueur tangible et l'image projetée sur le mur. Au contraire, l'interaction avec les ordinateurs de bureau est classiquement localisée en un point de l'espace qu'est l'ordinateur comme boîte grise.

En plus de cette distribution dans l'espace, une étude de la littérature montre que les systèmes de réalité mixte proposent le multiplexage spatial de l'interaction.

3.1.3.2. Multiplexage

Nous définissons d'abord le multiplexage et nous l'illustrons, avant de présenter ses enjeux de conception.

3.1.3.2.1. Définition

Le multiplexage en télécommunication est la communication de plusieurs signaux sur un canal commun. Dans le domaine de l'Interaction Homme-Machine, il s'agit de la façon d'utiliser plusieurs ressources d'interaction pour une seule tâche. Nous parlons de multiplexage spatial (respectivement temporel) si la tâche est effectuée grâce aux relations spatiales (respectivement temporelles) entre les ressources.
Par exemple, les interfaces graphiques traditionnelles offre un multiplexage temporel de l'interaction car la souris est utilisée pour toutes les tâches [Fitzmaurice, Buxton, 1997]. Pour effectuer une tâche, il faut donc activer les unes après les autres les ressources d'interaction qui se trouvent à l'écran. Ce phénomène est aussi expliqué dans [Beaudoin-Lafon, 2000] : Un unique dispositif physique, la souris, est utilisé pour réaliser toutes les tâches, et active plusieurs outils logiques, telle la barre de défilement, la commande de menu, le bouton de mise en forme, etc. Dans [Terrenghi, 2008], se trouve une classification de différents systèmes interactifs selon l'interaction (Tableau 1). Le résultat montre que parmi les systèmes étudiés, l'interaction caractérisée par le multiplexage spatial correspond plus souvent dans les systèmes de réalité mixte étudiés, alors que la distribution temporelle concerne les exemples d'interfaces graphiques.

Multiplexage temporel	**Multiplexage spatial**
Stylo lumineux sur tablet PC Souris Tablette graphique WACOM Touch Pad	MetaDesk [Ullmer et Ishii, 1997] HabilisDraw [Butter et St. Amant, 2004] Designer's Outpost [Klemmer et al., 2001] Props [Hinckley et al., 1994] Bricka [Fitzmaurice et al., 1995] Flatland [Mynatt et al., 1999] DataTiles [Rekimoto et al. 2001] ToolStone [Rekimoto et al. 2000] Navigation Blocks [Camarata et al., 2002] Illuminating Clay [Piper et al., 2002] ShapeTape [Balakrishnan et al. 1999]

Tableau 1 : Exemples de multiplexages spatiaux et temporels (d'après [Terrenghi, 2008]).

3.1.3.2.2. Enjeux de conception

Dans [Beaudoin-Lafon, 2000], il est question de la difficulté des concepteurs pour trouver l'équilibre entre multiplexage temporel et spatial. Ce problème existe déjà pour les concepteurs d'objets traditionnels: de la boîte à outils trop lourde et encombrante est né le couteau suisse, qui permet le multiplexage temporel. Mais la boîte reste la plus pratique lorsqu'on a besoin de changer d'outil souvent. De la même façon, les téléphones portables, appareils photo et baladeurs numériques qu'on utilisait souvent séparément sont maintenant des outils vendus fusionnés. Dans [Fitzmaurice, Buxton, 1997] une évaluation comparative a été menée. Les auteurs ont expérimenté les temps nécessaires pour réaliser des tâches avec une interface à multiplexage temporel (interface graphique)

versus une interface à multiplexage spatial (interface tangible). Entre un outil pour chaque tâche et un seul outil pour toutes les tâches, l'équilibre réside entre ces deux propositions :

- Avoir un instrument dédié pour une tâche permet de l'activer « presque » directement (presque car il faut un temps pour la prise en main : c'est l'opérateur *Homing*[10] de la méthode GOMS [Card et al., 1983] ; et il faut également savoir quel objet doit être utilisé pour quelle tâche).
- Activer avec la souris un instrument logique demande du temps, pour aller se positionner dessus.

Dans [Beaudoin-Lafon, 2000], l'auteur attire l'attention sur le fait que l'espace de conception est grand entre une souris unique et la table remplie de potentiomètres des ingénieurs du son. Entre les deux, il existe des solutions modulaires comme celle présentée dans [http://buglabs.net/products] qui semble être un compromis entre multiplexage spatial et temporel : les utilisateurs associent à un élément de base (Figure 27, à gauche) d'autres dispositifs comme une caméra (Figure 27, à droite), afin de construire un outil modulable.



Figure 27 : BUGbase et BUGCamera. Illustration extraite de http://buglabs.net/products.

Nous voyons donc que l'espace joue un nouveau rôle dans les systèmes de réalité mixte. L'interaction est diffusée dans l'espace et est davantage multiplexée spatialement.

Nous avons présenté l'utilisation de l'espace par les interfaces mixtes, après avoir montré la multiplication des possibilités de couplage entre physique et numérique et les difficultés de prototypage des systèmes de réalité mixte. Toutes ces questions concernent l'explosion des possibilités. Nous présentons maintenant les questions sur la liaison entre les mondes différents (physique et numérique).

3.2. Liaison entre des mondes différents

Malgré la multiplication des possibilités de couplages entre les mondes physique et numérique, ces mondes ne deviennent pas pour autant imbriqués de façon transparente. Nous les comparons à deux mondes régis par des lois différentes. Prolonger l'un par l'autre n'est pas une évidence. La combinaison à définir soulève des questions, comme la prise en compte du contexte dans l'interaction, les conflits entre physique et numérique ou comment garder les avantages des deux mondes en un seul système.

3.2.1. Prise en compte du contexte dans l'interaction

Les systèmes de réalité mixte sont une façon d'intégrer l'environnement des utilisateurs dans l'interaction. L'essence de ce paradigme est de mêler l'ordinateur au monde physique, et la prise en compte du contexte est une façon de mêler les deux mondes. L'interaction en contexte est un vaste axe de recherche, et le terme contexte ne connaît pas de définition consensuelle. Nous retenons celle de [Coutaz, Calvary, 2008]. Le contexte dépend de :

- L'utilisateur (par exemple un utilisateur aveugle ne peut pas utiliser l'écran),

[10] Retour à l'origine

- La plate-forme matérielle et logicielle sur laquelle il interagit (l'interaction sur un ordinateur de bureau n'est pas la même que sur un téléphone portable),
- L'environnement
- Social (l'utilisateur est-il seul ou entouré ?),
- Physique (par exemple dans Carcade, l'environnement physique constitue le décor du jeu et les obstacles que l'utilisateur doit éviter).

La prise en compte du contexte en interaction est une problématique qui n'est pas propre aux interfaces mixtes et concerne aussi l'informatique mobile et pervasive/ubiquitaire. De nombreuses questions découlent de la volonté de prendre en compte le contexte d'interaction.

- La difficulté conceptuelle : qu'est ce que le contexte ? Qu'est-il pertinent de prendre en compte et pour quelle situation ?
- La difficulté pratique de mise en œuvre : le problème technologique est grand. De nombreux travaux se sont penchés sur cette problématique [Rey, 2005][Salber et al., 1999].
- La difficulté d'exploitation du contexte : comment adapter les interfaces lorsque le contexte change ? [Coutaz, Calvary, 2008]

Bien qu'aujourd'hui, la prise en compte du contexte forme un axe de recherche en soi au sein du domaine de l'Interaction Homme-Machine, les systèmes de réalité mixte ont été parmi les premiers systèmes à prendre en compte le contexte. Par exemple, dans le DigitalDesk, l'arrangement spatial dans l'espace physique est obligatoirement pris en compte pour que le système fonctionne. Dans [Mackay, 1996], l'auteur questionne sur comment maintenir la correspondance entre l'information détectée dans le monde physique et la présentation de l'information numérique. Dans [Dourish, 2004], l'auteur invite les concepteurs à prendre le quotidien, disponible à portée de main, *where the actions is*[11] comme interface. Il s'agit là aussi d'inscrire l'interface dans le contexte.

3.2.2. Gestion des conflits entre les mondes physique et numérique

Quand l'accès à une ressource, qu'elle soit physique ou numérique, est partagé entre les mondes physique et numérique, il peut y avoir un conflit. Ce problème a été pris en compte dans [Mackay, 1996] [Dix et al., 2007]. Une ressource physique ou numérique peut être partagée entre les mondes physique et numérique de façon symétrique ou non. Elle est partagée de façon symétrique si le monde physique (utilisateur ou environnement) peut accéder à cette ressource comme peut y accéder que le système informatique.

Dans [Dix et al., 2007] est présenté l'exemple d'une ressource partagée de façon non symétrique : le bouton de programme de la machine à laver. L'utilisateur *et* le système peuvent agir sur le bouton du programme de la machine. Lorsque le système fait tourner ce bouton, cela signifie que le linge est en cours de nettoyage, servant alors de retour d'information sur l'avancement (du numérique vers le physique). Mais si l'utilisateur tourne ce bouton, cela ne signifie pas que le linge sera lavé plus vite. En effet, l'utilisateur s'en sert pour spécifier le début du programme à parcourir (du physique vers le numérique). Dans cet exemple, nous trouvons une ressource partagée, mais de façon non symétrique en entrée et en sortie. Dans [Mackay, 1996] nous trouvons un autre exemple de conflit non symétrique entre physique et numérique. Si un menu est projeté sur une feuille de papier (du numérique vers le physique), et qu'ensuite une autre feuille de papier recouvre la première, alors la projection se retrouve sur la seconde feuille de papier. Dans le monde physique la seconde feuille de papier aurait occulté le contenu de la première. De même dans le monde numérique des interfaces graphiques, lorsqu'une fenêtre recouvre une autre, elle l'occulte. Les ressources, comme les feuilles de papier ou le menu projeté, ne sont pas partagées entre le monde physique et le système. Il faut alors identifier chaque feuille de papier, comme proposé par [Mackay, 1998] à l'époque, ou utiliser *Second Light* [Izadi et al., 2008] qui permet de projeter des images différentes sur des supports qui se recouvrent.

Nous trouvons un exemple de ressource partagée de façon symétrique dans [Dix et al, 2007]. Le système de la bouilloire électrique peut s'éteindre seul. Dans ce cas, le système « pousse » le bouton

[11] où l'action se trouve

vers le bas (du numérique vers le physique). Mais l'utilisateur peut aussi pousser le bouton vers le haut (du physique vers le numérique). Dans cet exemple, une ressource est partagée de façon symétrique en entrée et en sortie. Il en découle une nécessité de résolution, si le système et l'utilisateur font une action antagoniste au même instant, par exemple par la prise en compte de priorités : c'est la plus forte des deux actions qui est prise en compte.
Ces exemples soulignent la nécessité d'étudier les solutions pour résoudre le conflit entre physique et numérique. Il convient donc d'identifier les différents types de situations dès les premières étapes de conception pour définir les cas de conflits entre physique et numérique.

3.2.3. Combinaison des avantages du physique et du numérique

Les avantages des deux mondes, physique et numérique, sont divers : fonction, rapidité, place nécessaire, satisfaction, plaisir, etc. Réduire le fossé entre physique et numérique signifie aussi interagir dans un monde qui prend le meilleur de chaque côté. Par exemple dans [Wellner, 1993], le DigitalDesk permet de combiner les intérêts des documents numériques et physiques dans un seul système. Avant, l'interaction avec les documents numériques se faisait avec le clavier, l'écran et la souris et l'interaction avec les documents physiques sur le bureau. Le lien entre les deux types de documents était limité à l'impression. Le but du DigitalDesk est de ne plus avoir besoin de choisir de façon définitive entre les deux. Il s'agit de minimiser le fossé entre pratique traditionnelle dans le monde physique et pratique avec l'informatique. Il convient de conserver les interactions traditionnelles avec le monde physique lorsque c'est souhaitable. Un système de réalité mixte, en augmentant un objet qui existe déjà, peut en garder les avantages. Dans l'exemple de Caméléon [Mackay, 1996], les contrôleurs ont besoin des bandes de papier pour se souvenir, communiquer et garder une trace : le garder dans la main, écrire dessus, décaler une bande de papiers sur le côté, ont une signification. Il est donc important de prendre en compte ces pratiques, voire de faire en sorte que l'activité soit continuée en cas de panne du système. De même dans [Dourish, 2004], l'auteur invite les concepteurs à prendre le quotidien, disponible à portée de main, *where the actions is*[12] comme interface. Il s'agit d'apporter le monde numérique dans le monde des utilisateurs plutôt que l'inverse. Ces conseils s'inscrivent dans l'intention de fournir aux utilisateurs les avantages du physique et du numérique dans un seul système.
Pourtant, afin de prendre en compte toutes les conséquences d'une telle demarche, [Mackay, 1996] préconise de considérer pas seulement les avantages, mais aussi les problèmes engendrés par la combinaison du physique et du numérique, et son influence sur leur activité.

Nous venons de présenter les questions de recherches liées à la liaison entre deux mondes différents, dont la prise en compte du contexte dans l'interaction, la gestion des conflits entre les mondes physique et numérique, et la combinaison des avantages du physique et du numérique. Nous avons vu que ces questions avaient été traitées dans la littérature, et qu'elles avaient des implications sur la conception. Nous présentons maintenant les questions liées aux aspects humains.

3.3. Aspects humains

Dans la littérature, les questions de recherche liées aux aspects humains concernent les activités principales et secondaires des utilisateurs du système, l'intuitivité des interfaces, le contrôle par l'utilisateur et le travail collaboratif, en groupe d'utilisateurs. Nous présentons ces questions dans cette partie.

[12] où l'action se trouve

3.3.1. Prise en compte des activités principales et secondaires

Malgré notre activité principale, comme travailler face à notre ordinateur, nous sommes réceptifs à d'autres signaux périphériques, comme la luminosité de la pièce, le bruit environnant, etc. Dès l'émergence des interfaces mixtes a été évoquée la question des activités principales et secondaires [Ishii, Ullmer, 1997]. Ces systèmes ont présenté une nouvelle opportunité pour prendre un compte nos différents niveaux d'implication dans une activité. Plusieurs niveaux d'observabilité certes existent dans les interfaces graphiques. Par exemple, une fenêtre d'avertissement peut bloquer le fil de l'interaction ou non (propriété d'insistance [Abowd et al., 1992]). Mais alors que la station de travail était au centre de l'attention de l'utilisateur, la diffusion dans l'espace physique des systèmes informatiques et la multiplication des possibilités de liaisons permet d'exploiter plus encore l'attention périphérique, en plus du centre principal d'attention (ou focus). Par exemple dans [Ishii, Ullmer, 1997] la diffusion de l'information numérique à la périphérie du centre d'attention de l'utilisateur est étudiée par l'utilisation de la lumière, du son, du vent ou de l'eau. Les interfaces mixtes peuvent utiliser ce nouvel espace physique, ces nouvelles possibilités de liens entre physique et numérique, ainsi que le contexte, pour prendre en compte et s'adapter aux activités principales et secondaires des utilisateurs.

3.3.2. Recherche d'interfaces intuitives et naturelles

Un mythe propre aux interfaces tangibles serait qu'elles sont plus intuitives, naturelles ou réalistes. Cette idée vient du fait que le développement intellectuel et le développement moteur sont liés. Les enfants jouent et apprennent via le toucher [Dix, 2007]. L'interaction avec le monde physique et tangible est décrite comme la source de compréhension du monde [Dourish, 2004].
Pourtant, l'exemple des détecteurs de mouvements ou de présence utilisés dans des toilettes pour le robinet d'eau, le séchoir pour les mains ou pour la porte dans certains trains express régionaux de la SNCF montre que quand la technologie est presque invisible, l'interaction ne se révèle pas elle-même. Si l'utilisateur ne voit aucun dispositif pour la capture et que toutes les actions sont possibles, alors il ne sait pas forcément quoi faire. En effet, plusieurs expériences montrent que l'intuitivité des interfaces mixtes doit être remise en question. Une correspondance directe entre le monde physique et le monde numérique n'est pas forcément plus simple ou intuitive, voire appréciée par les utilisateurs. Un premier exemple d'expérience est décrit dans [Johnson et al., 2002]. Trente-trois personnes, 17 expertes (11 hommes et 7 femmes) et 16 novices (6 hommes et 10 femmes), ont joué à un jeu de snowboard (SSX) et de tir (Time Crisis 2) avec des contrôleurs classiques (types manettes), et des contrôleurs nouveaux en formes de snowboard et d'arme à feu. L'ordre des jeux et des contrôleurs a été contrebalancé selon les niveaux d'expérience et le genre des sujets, et ces derniers ont répondu à un questionnaire et à un entretien. Au final, le jeu de tir était plus facile à contrôler et plus intuitif avec le contrôleur en forme d'arme à feu qu'avec la manette, et les utilisateurs l'ont trouvé réaliste (c'est-à-dire qu'il simulait de façon satisfaisante le comportement équivalent dans la vie courante ou « réelle »). Mais au contraire, le jeu de snowboard avec le contrôleur en forme de snowboard était plus difficile à contrôler, moins intuitif, et moins réaliste. Donc un contrôleur imitant l'activité traditionnelle n'est pas nécessairement plus réaliste ou même plus intuitif qu'un contrôleur classique.
Au-delà même de l'interaction intuitive et naturelle, dans cette précédente étude, les contrôleurs nouveaux en formes de snowboard et d'arme à feu ont été reportés comme plus amusants que les manettes de jeu standard. Néanmoins, ce dernier constat ne peut être généralisé. Ainsi dans [Terrenghi, 2008b] une étude explore la différence en deux interfaces équivalentes, l'une en deux dimensions (avec des objets projetés sur une surface captant le toucher) et l'autre en 3D (avec des objets tangibles). Les résultats montrent que les utilisateurs ont trouvé l'interaction avec l'interface 2D davantage amusante.

Une approche de conception pour aller vers l'intuitivité des interfaces est donc plutôt d'étudier ce que font les utilisateurs intuitivement avant de concevoir. Dans [Dourish, 2004][Mackay, 1996], il est ainsi préconisé d'étudier les pratiques de l'utilisateur afin d'insérer des fonctions numérique dans les pratiques existantes. D'autres approaches ont été proposées pour provoquer la bonne intuition chez l'utilisateur, comme l'étude de l'affordance des objets [Norman, 1999], ou l'utilisation de métaphores avec le monde réel [Fishkin, 2004]. Ainsi, un objet qui doit être déplacé sur une table peut être plat et lourd afin provoquer la bonne utilisation par son affordance. Un outil pour mesurer peut prendre la forme d'une règle, afin de provoquer la bonne intuition chez l'utilisateur grâce à une métaphore. Pourtant, cette dernière solution n'est pas universelle et n'est adaptée qu'aux utilisateurs dont la règle fait partie de la culture.

Nous présentons maintenant une autre question liée aux aspects humains : la place donnée dans les systèmes de réalité mixte pour la conscience, la connaissance et le contrôle par l'utilisateur.

3.3.3. Place pour la conscience, la connaissance et le contrôle par l'utilisateur

Le mélange entre physique et numérique pourrait, à terme, semer la confusion chez l'utilisateur [Mackay, 1996] : comment faire la différence entre le physique et le numérique ? De même dans [Greenfield, 2006] se trouve la question du contrôle par l'utilisateur. Une opposition émerge entre l'environnement augmenté, où l'utilisateur a le contrôle, et un environnement intelligent.
Pour permettre à l'utilisateur de maîtriser le système, certains auteurs préconisent d'appliquer la notion de *seamful design*[13] [Bell et al., 2006] [Weiser, 1994], par opposition au terme *seamless*[14][Weiser, 1994]. Il est possible de rentre visible les limitations des technologies pour enrichir l'expérience de l'utilisateur, plutôt que de les dissimuler. Ainsi dans un jeu mobile [Barkhuus, 2005], les utilisateurs ont exploité les imprécisions du G.P.S. pour piéger leurs adversaires dans le jeu [Barkhuus, 2005], car l'interface permettait à l'utilisateur de maîtriser le système.
La question de la maîtrise du système par l'utilisateur a été exploitée artistiquement, par exemple dans l'installation *Lies, all lies*[15] (2003) de l'artiste Hannes Nehls [Nehls]. Cette installation révèle au destinataire (utilisateur) une différence subtile entre physique et numérique. L'utilisateur du système prend l'ombre projetée sur le mur pour *son* ombre (Figure 28), jusqu'à ce qu'elle se comporte de façon inattendue et incontrôlable (Figure 28), car générée par le système et substituée à l'ombre physique réelle.



Figure 28 : *Lies, all lies*, Hannes Nehls, 2003. Illustration extraite de[Nehls].

La dernière question de recherche sur les aspects humains est la question sur le travail en groupe, que nous présentons maintenant.

3.3.4. Favorisation du travail en groupe

Déjà dans les travaux séminaux sur les interfaces mixtes, la réalité augmentée ou les interfaces tangibles, se trouvent des références aux aspects multi-utilisateurs. Les premiers systèmes informatiques pour le travail collaboratif sont une extension des interfaces graphiques [Brave et al., 1998], par exemple en fournissant un accès partagé à des ressources en ligne ou un environnement

[13] Conception avec coutures apparentes
[14] Sans couture
[15] Mensonges

numérique, ou en proposant l'usage de la vidéo-conférence quand un contact direct est nécessaire. Pourtant, le monde physique et notre interaction avec lui joue un rôle important dans notre compréhension du monde ainsi que dans les relations sociales [Dourish, 2004] : l'incarnation physique est importante. Pour respecter cet aspect, nous trouvons des systèmes qui proposent aux utilisateurs multiples d'interagir de façon plus directe et en étant plus ancré dans le monde physique grâce à une interface tangible [Brave et al., 1998]. En effet, les objets physiques joue un rôle important dans le travail collaboratif [Mackay, 1998], et une informatisation de ce travail qui se débarrasse de ces objets risque de perdre des éléments importants pour l'interface. Depuis, des prototypes essayant de tirer parti du monde physique existant et des normes sociales ont été proposés, plutôt que de se baser sur les systèmes informatiques existants [Kanis et al., 2005].

4. Synthèse

4.1. Motivations

Nous avons proposé une définition des interfaces mixtes et rappelé la terminologie utilisée. Nous avons ensuite présenté des exemples illustrant la variété du domaine. Nous avons souligné les spécificités des interfaces mixtes, mais également les aspects que ce paradigme d'interaction partage avec d'autres, en identifiant des grands défis liés à ces interfaces. Les objectifs de nos travaux découlent de ces défis :

- L'explosion des possibilités : Nous visons la prise en compte de la multitude et variabilité des couplages possibles entre monde physique et numérique lors de la conception.
- Le prototypage : Nous nous intéressons à la conception et réalisation d'outils pour faciliter cette activité, en l'insérant dans la conception.
- L'utilisation de l'espace : Nous visons la caractérisation spatiale (et temporelle) des interfaces mixtes, puisque leur utilisation de l'espace a été reconnue comme particulière.
- La gestion des conflits entre physique et numérique : Nous souhaitons mettre en exergue lors de la conception les éléments d'une interface qui peuvent être en situation de conflit entre le physique et le numérique.

Les autres aspects identifiés ne sont pas nécessairement traités explicitement dans nos travaux, bien qu'ils y soient présents, puisque partie intégrante des systèmes de réalité mixte.

La motivation de nos travaux s'inscrit non seulement dans ce contexte de recherche en Interaction Homme-Machine, mais aussi dans son illustration pratique lors de la conception concrète de systèmes de réalité mixte. De nombreux systèmes de réalité mixte ont été conçus et développés dans les dernières décennies, et nous constatons qu'il est encore difficile de les comparer. Il est également difficile d'explorer facilement les alternatives de conception. Par exemple, au sein de notre équipe de recherche sur l'ingénierie de l'Interaction Homme-Machine du Laboratoire d'Informatique de Grenoble (LIG), nous avons développé de nombreux systèmes de réalité mixte. Ces systèmes ont souvent été développés de façon ad hoc, sans garder de trace du processus de conception. Par conséquent, la conception du système suivant devait être reprise du début, et nous rencontrions les mêmes problèmes. Nous n'étions pas capables d'explorer l'ensemble de l'espace de conception de façon systématique, et souvent nous trouvions une meilleure solution une fois le développement terminé. Nous avons également rencontré des problèmes de compréhension lors de l'explication de choix de conception. En tant que concepteurs d'interfaces mixtes, nous sommes personnellement concernés par ces problèmes.

4.2. Objectifs

Dans ce contexte, nous visons l'aide à la conception et à la réalisation de prototypes d'interfaces mixtes. Notre objectif global est de comprendre, de cerner, et de contribuer à l'espace de conception des systèmes de réalité mixte.

Afin d'aider la conception des systèmes mixtes, nos travaux visent à définir un modèle d'interaction [Appert, 2007][Beaudoin-Lafon, 2004] uniforme capitalisant les travaux existants sur les interfaces mixtes, pour obtenir une compréhension globale de ces interfaces. Nous soulignons le fait que notre objectif vise à unifier les études concernant la conception et le développement. Notre but n'est pas de définir encore un autre modèle ou encore un autre outil logiciel. En utilisant les résultats de nos travaux, un concepteur doit savoir comment utiliser les travaux conceptuels existants de façon systématique pendant la conception.
De plus, comme souligné dans [Beaudoin-Lafon, 2004], les nouveaux paradigmes d'interaction nécessitent à la fois des modèles d'interaction et des outils correspondants afin d'en faciliter le prototypage. Tout naturellement donc, afin de rendre opérationnel notre modèle d'interaction, un deuxième objectif de nos travaux est l'étude des outils de développement et de prototypage à la lumière de notre modèle d'interaction. Les développeurs doivent être capables de produire des prototypes ou ébauches, basés sur les concepts clés de notre espace de conception, tout en profitant des avantages des outils de prototypage existants.

4.3. Démarche scientifique

Même si nous ne traitons pas les phases de développement et d'évaluation, nos travaux couvrent la phase de conception et la phase de maquettage/prototypage des interfaces mixtes. Pour cela, nous nous sommes appuyés sur une démarche à la fois conceptuelle (cadre du haut de la Figure 29) et expérimentale (cadre du bas de la Figure 29).

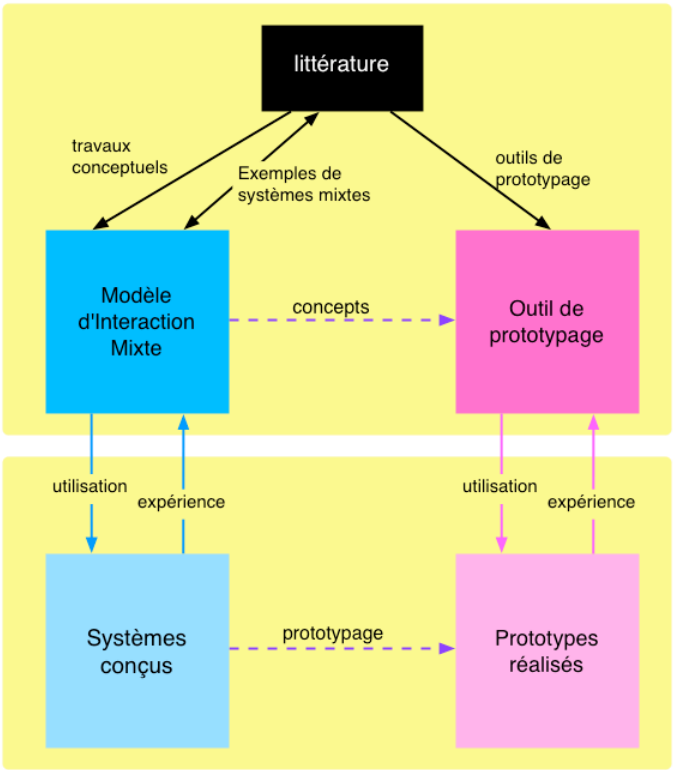


Figure 29: Démarche scientifique à la fois conceptuelle et expérimentale.

Pour construire le modèle d'interaction mixte (Figure 29, flèches entrantes dans le cadre bleu foncé) :

1. Nous avons étudié les travaux conceptuels pour identifier les parties redondantes ou manquantes parmi les outils conceptuels de la littérature, et les notions utiles à la conception qui devaient d'être capitalisées à l'intérieur d'un outil de conception.
2. Nous avons considéré les exemples issus de la littérature pour identifier les requis.
3. Nous avons basé la construction du modèle d'interaction mixte sur notre propre expérience de conception de systèmes de réalité mixte.

Pour évaluer et améliorer le modèle d'interaction mixte (Figure 29, flèches pleines sortantes du cadre bleu foncé) :

1. Nous avons appliqué le modèle pour décrire les exemples de la littérature.
2. Nous avons utilisé le modèle pour concevoir des systèmes de réalité mixte.

Pour l'utilisation et l'évaluation du modèle, nous avons travaillé avec des professions actrices de la conception des interfaces mixtes : informaticiens, designers, et ergonomes. De plus, nous avons complété ces études empiriques (fondées sur l'observation et l'expérience) par des méthodes conceptuelles complémentaires, comme un questionnaire basé sur l'approche de [Anranda et al., 2007], ou des validations conceptuelles basées sur les approches de [Olsen, 2007] ou [Blackwell, Green, 2000].

Pour construire l'outil de prototypage (Figure 29, flèches entrantes dans le cadre rose foncé) :

1. Nous avons étudié les outils de prototypage de la littérature.
2. Nous avons exploité les concepts du modèle d'interaction.
3. Nous nous sommes aussi basés sur notre expérience de prototypage de systèmes de réalité mixte.

Pour évaluer et améliorer l'outil de prototypage, (Figure 29, flèches sortantes du cadre rose foncé), nous avons prototypé les systèmes conçus. Pour cela, nous avons travaillé avec des informaticiens et des designers. Nous avons complété ces études empiriques par des expériences en groupe focalisé et par une méthode de validation conceptuelle issue de [Olsen, 2007].

Tout en respectant cette démarche, notre approche a été incrémentale : nous avons construit plusieurs versions du modèle d'interaction mixte et plusieurs versions de l'outil de prototypage, en intégrant à chaque itération de nouveaux résultats de notre démarche conceptuelle (cadre du haut de la Figure 29) et expérimentale (cadre du bas de la Figure 29). Concrètement, nous avons fait des allers-retours entre les différents éléments de la Figure 29 pour enrichir notre modèle d'interaction mixte et notre outil de prototypage à chaque itération.

Partie I : Conception

Chapitre 3 : État de l’art des outils conceptuels

Ce chapitre présente un état de l'art et une analyse des outils conceptuels existants pour la conception. Nous avons présenté dans le chapitre précédent notre objectif d'aide à la conception des systèmes de réalité mixte. Notre objectif global est de comprendre, de cerner, et de contribuer à l'espace de conception des interfaces mixtes en définissant un modèle d'interaction uniforme pour les interfaces mixtes. Nous avons souligné que notre démarche s'appuie sur l'étude des travaux existants, dans un souci de capitalisation. Dans ce chapitre, nous présentons un état de l'art des outils conceptuels pour la conception des systèmes de réalité mixte. Chaque outil conceptuel que nous présentons dans ce chapitre est étudié sous l'angle des enjeux recherche liés aux interfaces mixtes, présentés au Chapitre 2 :

- L'explosion des possibilités, avec :
- La multiplication des possibilités de couplage entre physique et numérique,
- La multiplication des difficultés pour prototyper,
- L'utilisation de l'espace ;
- La liaison entre des mondes différents, avec :
- La prise en compte du contexte dans l'interaction,
- La gestion des conflits entre les mondes physiques et numériques,
- La combinaison des avantages du physique et du numérique ;
- Les aspects humains, avec :
- La prise en compte des activités principales et secondaires,
- La recherche d'interfaces intuitives et naturelles,
- La place pour la conscience, la connaissance et le contrôle par l'utilisateur,
- La favorisation du travail en groupe.

Tout au long de ce chapitre nous illustrons les outils conceptuels sur l'exemple du système de chirurgie augmentée CASPER, introduit à la section 2.1.2.2 du Chapitre 2. Nous rappelons que le système CASPER [Dubois, 2001] permet l'acquisition et la modélisation préopératoires de la région d'une effusion péricardique à partir de laquelle une trajectoire idéale de l'aiguille de ponction est planifiée. Pendant l'opération, le chirurgien est guidé grâce à un localisateur optique qui capture la position de l'aiguille en trois dimensions. La Figure 30 montre l'application pendant l'intervention (étape de guidage). Sur l'écran, la position et l'orientation de l'aiguille sont représentées par deux croix tandis qu'une croix fixe représente la trajectoire idéale (Figure 31). Quand les trois croix sont superposées, la trajectoire de l'aiguille correspond à la trajectoire idéale.



Figure 30 : Système CASPER, première version (d'après [Dubois, 2001]).

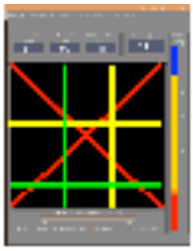


Figure 31 : Visualisation des trajectoires de l'aiguille de ponction sur l'écran du système CASPER, première version (d'après [Dubois, 2001]).

Nous présentons ensuite une analyse de la contribution des travaux conceptuels presentés au regard de nos objectifs. Nous montrons quelles sont les dimensions utiles à la conception qu'ils apportent au domaine, ainsi que leurs couvertures, leurs chevauchements et leurs manques.

1. Tour d'horizon des outils conceptuels

Les tentatives de conceptualisation des interfaces mixtes sont nombreuses. Très tôt, les travaux faisant explicitement référence aux interfaces tangibles (que nous avons définies à la section 1 du Chapitre 2) se sont démarqués comme une classe à part et ont suscité beaucoup d'intérêt. Nous proposons dans cette partie un état de l'art des outils conceptuels, en commençant d'abord par les travaux les plus généraux sur les systèmes de réalité mixte, pour terminer par ceux qui se réclament des interfaces tangibles.

1.1. Outils conceptuels pour les interfaces mixtes

1.1.1. Interaction basée sur la réalité

La notion de *Reality Based Interaction*[16] [Jacob et al., 2008] propose un cadre d'analyse uniforme des paradigmes d'interaction émergents. Ces paradigmes considérés sont la réalité virtuelle, mixte ou augmentée, les interfaces tangibles, l'informatique ubiquitaire et pervasif, sensible au contexte, mobile, ou encore les modalités d'interaction dites passives. Les auteurs montrent que ces paradigmes partagent une utilisation des capacités humaines développées en lien avec :

1. le monde physique (la gravité par exemple),
2. le corps,
3. l'environnement physique,
4. l'environnement social.

Par exemple, la résistance des matériaux est une notion qui vient de la physique et qu'utilisent les interfaces tangibles basées sur les contraintes et des interfaces mobiles portées dans la main comme [Williamson et al., 2007].

Pour autant, l'intérêt des systèmes informatiques réside dans le fait qu'ils ne sont pas une imitation fidèle de la réalité, mais nous permettent de faire des choses impossibles dans le monde physique [Dourish, 2004]. L'exploitation du monde physique, du corps, de l'environnement physique et social doit donc être pondéré lors de la conception d'un système. Pour aider les concepteurs, les auteurs proposent de considérer les facteurs suivants qui peuvent ne pas être en accord avec l'exploitation du monde physique, du corps, de l'environnement physique et social :

- le pouvoir d'expression (dans un domaine d'application donné, les utilisateurs peuvent réaliser un grand nombre de tâches),
- l'efficacité (les utilisateurs peuvent réaliser une tâche rapidement),
- la versatilité (le système peut être utilisé pour différents domaines d'application),
- l'ergonomie (les utilisateurs peuvent réaliser une tâche avec confort et en sécurité),
- l'accessibilité (des utilisateurs avec capacités physiques différentes peuvent réaliser une tâche),
- la facilité de développement et de production.

Il s'agit pour les concepteurs de pondérer l'exploitation du monde physique, du corps, de l'environnement physique et social avec ces 6 aspects. Dans l'exemple de CASPER, les concepteurs ont fait un compromis sur l'exploitation du monde physique, du corps, de l'environnement physique et social pour atteindre une meilleure ergonomie : en augmentant l'évaluation, le système permet au chirurgien d'opérer avec plus de sécurité.

Parmi les enjeux liés aux systèmes de réalité mixte que nous avons présentés au Chapitre 2, ce travail met l'accent sur la combinaison des avantages du physique et du numérique (Chapitre 2, section 3.2.3) et sur la prise en compte du contexte dans l'interaction (Chapitre 2, section 3.2.1).

[16] Interaction basée sur la réalité

1.1.2. Continuum de Milgram

Dans [Milgram, Kishino, 1994] se trouve une définition de la réalité mixte comme le continuum virtualité-réalité (Figure 32). Entre les deux extrema, les environnements réels et les environnements virtuels, nous trouvons la réalité augmentée et la virtualité augmentée.

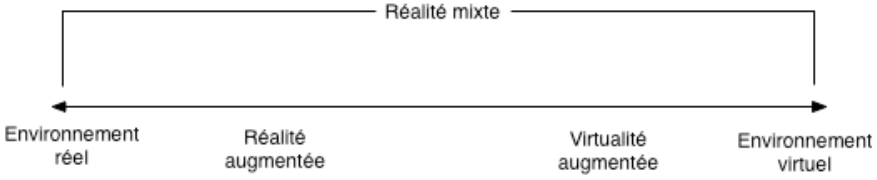


Figure 32 : continuum de Milgram.

Par exemple, le système CASPER est un système de réalité augmentée. Cette classification concerne les enjeux liés à l'explosion des possibilités liés aux systèmes de réalité mixte que nous avons présentés à la section 1 du Chapitre 2. Elle propose de caractériser la manière dont les mondes physiques et numériques sont liés. Cette classification reste très générale et elle est par conséquent difficilement utile pour la conception.

1.1.3. Taxonomie de Mackay

Dans [Mackay, 1996], l'auteur introduit une classification des interfaces mixtes basée sur la cible des augmentations. En effet, les augmentations ciblent différentes parties de l'interaction, comme :

- Les utilisateurs : c'est le cas par exemple des lunettes semi-transparentes que porte l'utilisateur.
- Les objets d'interaction : c'est le cas par exemple d'objets équipés d'un petit écran, ou de tout autre effecteur ou capteur.
- L'environnement : c'est le cas par exemple du projecteur qui permet d'afficher une information sur un mur.

Par exemple, dans la version du système CASPER présentée, le système augmente un objet d'interaction (l'aiguille de ponction) et l'environnement (grâce aux écrans de visualisation). Au contraire, dans la seconde version du système, le chirurgien (cible de l'augmentation) portait des lunettes semi-transparentes.

Cette taxonomie concerne les enjeux liés à l'explosion des possibilités liés aux systèmes de réalité mixte que nous avons présentés à la section 3.1 du Chapitre 2. Elle a constitué un premier pas vers une meilleure compréhension des nouvelles possibilités de liens entre mondes physique et numérique et de l'espace utilisé pour l'interaction. D'autres travaux sont venus compléter cette première approche.

1.1.4. Critères ergonomiques de Bach et Scapin

Les critères ergonomiques sont conçus pour l'inspection des interfaces afin d'améliorer leur utilisabilité. Ils peuvent servir aussi de façon prédictive, lors de la conception d'une interface, ce qui nous intéresse ici.

Parmi l'ensemble des propositions de critères ergonomiques, nous trouvons ceux de [Abowd et al., 1992], [Nielsen, 1994] et [Bastien, Scapin, 1993]. Ces derniers ont été étendus et revus dans [Bach, Scapin, 2005] afin de prendre en compte les « Environnements Virtuels », c'est-à-dire les interfaces de réalité virtuelle (définies au Chapitre 2, section 1). Même si dès le premier chapitre de cette thèse, nous nous sommes démarqués de ce paradigme d'interaction, ces travaux peuvent apporter une perspective ergonomique à la caractérisation des interfaces mixtes.

Nous considérons d'abord le critère de compatibilité. La compatibilité est l'accord pouvant exister entre les caractéristiques des utilisateurs (mémoire, perceptions, anatomie, habitudes, compétences, âge, attentes, etc.) et des tâches, d'une part, et l'organisation des sorties, des entrées et du dialogue d'une application donnée, d'autre part. Les exemples de recommandations liées à ce critère ergonomique soulignent que ce critère aborde plusieurs enjeux des interfaces mixtes, dont celui de

l'utilisation de l'espace (présenté à la section 3.1.3 du Chapitre 2) : « Si une tâche doit être effectuée à un endroit spécifique, alors il est nécessaire que les objets utiles à la réalisation de cette tâche se trouvent à cet endroit ».
Nous considérons maintenant les critères des actions explicites et du contrôle utilisateur. Par « actions explicites », les auteurs entendent que les actions de l'utilisateur sur le système doivent déclencher l'exécution de l'opération demandée, et pas d'autres, et ce, au moment où il le demande. Par « contrôle utilisateur », les auteurs entendent le fait que l'utilisateur doit toujours avoir la main, il doit contrôler le déroulement (ex.: interrompre, reprendre) des traitements informatiques en cours. Parmi les enjeux liés aux systèmes de réalité mixte que nous avons présentés au Chapitre 2, ces critères ergonomiques mettent l'accent sur les enjeux liés à la place pour la conscience, la connaissance et le contrôle par l'utilisateur (Chapitre 2, section 3.3.3).
D'autres critères mettent l'accent sur d'autres enjeux, comme le critère « signifiance des codes, dénominations et comportements » qui met l'accent sur la recherche d'interfaces intuitives et naturelles (Chapitre 2, section 3.3.2) et le critère d'« homogénéité/cohérence » qui met l'accent sur la prise en compte du contexte dans l'interaction (Chapitre 2, section 3.2.1).

1.1.5. Caractérisation des liens entre les mondes physique et numérique

L'objet central d'étude dans [Renevier, 2004] est le lien entre les mondes physiques et numériques, dans le cas particulier des systèmes de réalité mixte collaboratifs et mobiles.
Tout d'abord, la création des liens entre physique et numérique y est caractérisée. Lors de sa création, le **mode d'interaction** avec le lien peut être :

- Passif : le lien entre éléments physiques et éléments numériques est automatiquement déterminé par le système (position, nature, rendu, etc.). C'est l'exemple de l'aiguille de ponction péricardiale dans CASPER [Dubois, 2001].
- Actif : l'utilisateur spécifie une partie ou la totalité des données qui définissent le lien. C'est l'exemple des associations entre les objets et les informations dans [Carvey et al., 2006] où l'utilisateur associe ce qu'il veut via une balance USB.

Ensuite, en plus de sa création, l'accès, la modification et la destruction du lien sont aussi caractérisés par le mode d'interaction. La modification et la destruction du lien peuvent être passif (*Push*) ou actif (*Pull*).

Ensuite, la **temporalité** des liens est caractérisée : un lien peut être éphémère ou persistant.

Nous trouvons également une caractérisation de l'interaction dans l'espace, via la **localisation** des liens par rapport aux utilisateurs. Nous trouvons trois cas de figures :

- Les liens sont co-localisés, les utilisateurs se déplacent autour. C'est par exemple le cas de *The Golden Calf* [Shaw, 1994] (section 2.3.1 du Chapitre 2) où l'utilisateur se déplace et déplace l'écran autour du socle, pour regarder le veau d'or depuis différents points de vue.
- Les utilisateurs sont co-localisés, les liens n'existent alors que lorsque les utilisateurs sont réunis à cet endroit. C'est le cas par exemple d'Alexitimia [Gaetano Adi, 2006] (section 2.3.4 du Chapitre 2) où l'objet transpire quand les utilisateurs le touchent.
- Les liens sont repartis dans tout l'espace d'utilisation. C'est le cas de l'ARQuake [Thomas et al., 2000] que nous avons présenté à la section 2.2.1.2 du Chapitre 2. Les liens sont repartis sur le terrain de jeu et les joueurs se déplacent sur ce terrain.

Enfin, trois **types d'interaction** sont caractérisés pour les systèmes de réalité mixte collaboratifs et mobiles :

- Les utilisateurs interagissent avec des objets à travers des liens. C'est l'exemple de Tuttuki Bako (http://www.asovision.com/tuttuki/) présenté à la section 2.2.2.2 du Chapitre 2, où l'utilisateur crée le lien en introduisant son doigt dans la boîte pour interagir avec les objets affichés.

- Les objets interagissent avec les utilisateurs à travers des liens. C'est l'exemple de Carcade [Fischer, Luge, Polk, 2008] présenté à la section 2.2.1.1 du Chapitre 2, où l'utilisateur perçoit les objets du décor environnant dans le jeu.
- Les liens servent à l'échange entre utilisateurs : il s'agit alors de communication homme-homme médiatisée et la fonction des liens est d'établir une communication entre plusieurs utilisateurs. C'est par exemple le cas de *Voice Boxes* [Jeremijenko, 1995] présenté à la section 2.3.3 du Chapitre 2, où les utilisateurs peuvent communiquer de façon asynchrone grâce aux messages enregistrés dans les boîtes.

Parmi les enjeux liés aux interfaces mixtes que nous avons présentés au Chapitre 2, ces caractéristiques mettent l'accent sur les enjeux liés à l'utilisation de l'espace (section 3.1.3) et à la place pour le contrôle par l'utilisateur (section 3.3.3).
En résumé, les caractéristiques des liens entre mondes physique et numérique sont utiles en conception. Ces caractéristiques complètent les précédentes (Milgram, Mackay) et pourraient aussi compléter les travaux que nous présentons maintenant.

1.1.6. Modèle spatial

La modélisation de l'espace présenté dans [Benford, Fahlen, 1993] a été conçue pour l'interaction entre membres d'un groupe dans un environnement virtuel. Nous appliquons ici ce cadre d'étude pour caractériser les relations spatiales entres utilisateurs et/ou objets d'interaction. Les auteurs introduisent les notions de :

- **Medium** : Il s'agit du moyen utilisé par une entité pour communiquer, comme l'audio ou le texte par exemple.
- **Aura** : Il s'agit d'un sous-ensemble de l'espace dans lequel une entité peut communiquer. L'aura est dépendante du medium. Par exemple dans certaines situations, il est possible de voir plus loin qu'on n'entend. L'aura visuelle est alors plus grande que l'aura auditive.
- **Focus** : Il s'agit de l'espace où est dirigée l'attention d'une entité : il s'agit par exemple du champ de vision.
- **Nimbus** : Il s'agit de l'espace où est dirigée la présence d'une entité : par exemple l'espace d'où l'entité peut être vue.
- ***Awareness***[17] : La conscience qu'une entité a d'une autre est définie par le focus et le nimbus.
- Plus une entité E1 est dans le focus d'une entité E2, plus E2 est consciente de E1.
- Plus une entité E1 est dans le nimbus d'une entité E2, plus E1 est consciente de E2.
- Ce n'est pas une relation symétrique entre entités. La Figure 33 montre un exemple de configuration spatiale d'après [Benford, Fahlen, 1993].
- **Adaptateurs** : Il s'agit d'un objet capable de modifier (c'est-à-dire amplifier ou atténuer) l'aura, le focus ou le nimbus d'une entité. Par exemple, un microphone amplifie l'aura d'un utilisateur pour le medium sonore.

Cette étude fournit un cadre de caractérisation des relations spatiales entre utilisateurs et objets d'interaction, ce qui permet au concepteur d'avoir une meilleure appréhension de la diffusion de l'interaction dans l'espace opérée par le système de réalité mixte. En cela, elle concerne plus particulièrement l'enjeu lié à l'utilisation de l'espace présenté à la section 3.1.3 du Chapitre 2.

[17] Conscience

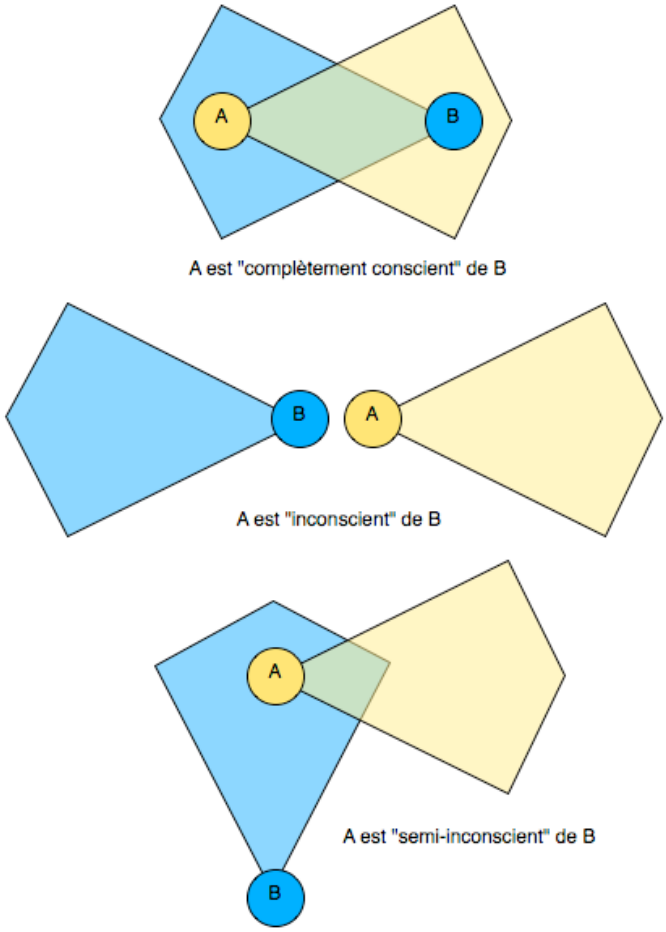


Figure 33: Exemple de configurations spatiales avec les focus de deux entités A et B (d'après [Benford, Fahlen, 1993]). A et B sont des entités et les quadrilatères de mêmes couleurs représentent l'espace ils portent leur attention (focus).

1.1.7. Mouvements attendus, captés et souhaités

Dans [Benford et al., 2005] un cadre de conception basé sur l'étude des mouvements de l'utilisateur est présenté selon trois perspectives :

- Les **mouvements attendus** : il s'agit d'identifier les mouvements naturels pour un utilisateur donné avec une interface donnée dans un environnement donné. Les auteurs proposent également d'identifier les mouvements moins naturels, voire impossibles. Ces mouvements ont plusieurs propriétés, comme :
    - Les **degrés de liberté** : Quelles enchaînements de translation et rotations peuvent être attendus ?
    - L'**intervalle** : Jusqu'où peut aller le mouvement pour chaque degré de liberté ?
    - Leur **vitesse** pour chaque degré de liberté,
    - Leur précision,
    - Leur stabilité et maintenabilité.

  Ces mouvements attendus dépendent de l'objet avec lequel l'utilisateur interagit (sa forme ou son poids, par exemple). Ils dépendent aussi de l'environnement. Par exemple dans CASPER, le concepteur ne peut attendre du chirurgien que les mouvements nécessaires à l'opération chirurgicale.
- Les **mouvements capturés** sont ceux qui peuvent être mesurés par le système, grâce à des capteurs. Ils ont les mêmes propriétés que les mouvements attendus. En revanche on peut aussi les caractériser en plus par leur **coût** (consommation des ressources matérielles, temps de développement, etc.), les **conditions environnementales** dont ils ont besoin (par exemple

la luminosité pour les mouvements captés par une caméra), les **ressources** du processeur qu'ils demandent (compromis entre précision et latence).

- Les **mouvements souhaités** sont ceux dont l'application a besoin. Par exemple avec CASPER, l'application a besoin de quatre nombres représentant l'orientation et la position en trois dimensions de l'aiguille de ponction manipulée par le chirurgien.

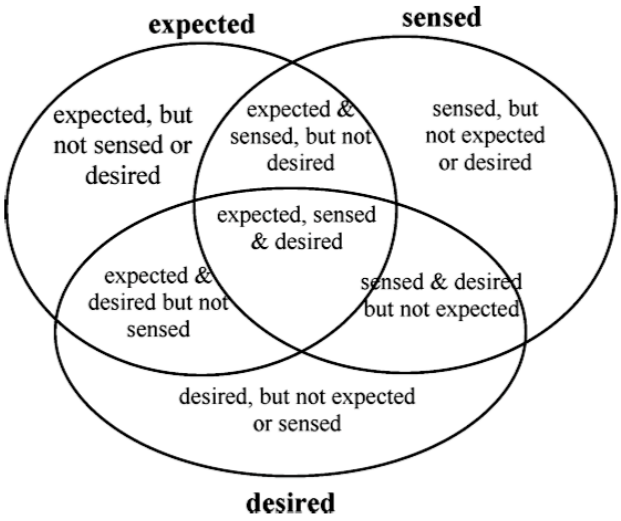


Figure 34: Classes de mouvements. Illustration extraite de [Benford et al., 2005].

Ce cadre d'étude permet d'étudier les intersections entre ces différentes classes de mouvements comme le montre la Figure 34. Les auteurs affirment que certaines parties de cet espace de conception sont négligées. Par exemple, les mouvements attendus mais pas capturés peuvent être étudiés davantage par les concepteurs d'interfaces. Puisque ce sont des mouvements attendus, il est possible par exemple de :

- Augmenter l'intervalle de capture pour qu'ils soient capturés,
- Contraindre les mouvements pour que ceux qui ne peuvent pas être capturés ne soient pas attendus,
- Prévenir l'utilisateur qu'il sort de l'intervalle de capture (observabilité [Abowd et al., 1992]).

Les auteurs montrent que ces classes de mouvements peu étudiées ne sont pas forcément des problèmes mais peuvent être aussi des opportunités pour l'utilisateur de sortir délibérément de l'intervalle de capture. C'est par exemple le cas du jeu présenté dans [Barkhuus, 2005] où les utilisateurs ont exploité les zones non couvertes par le G.P.S. pour se cacher dans le jeu. Certains des mouvements attendus de la part de l'utilisateur ne doivent pas être destinés au système, et donc ne doivent pas être captés. C'est l'exemple des mouvements, qui, avec la Wiimote de Nintendo, sont tout de même captés et pris en compte dans le système même si ce n'était pas l'intention de l'utilisateur.

Ces travaux sur les mouvements forme un cadre pour la conception, qui traite notamment les enjeux présentés au Chapitre 2 et liés à la place laissée à la conscience, la connaissance et au contrôle par l'utilisateur (Chapitre 2, section 3.3.3) et à l'intuitivité des interfaces (Chapitre 2, section 3.3.2).

1.1.8. ASUR

ASUR (Adaptateur, Système, Utilisateur, entités Réelles) est une notation de conception qui a été introduite dans [Dubois et al., 1999], puis étendue [Dubois, Gray, 2007]. L'interaction au sein d'un système de réalité mixte est modélisée selon ASUR par quatre *composants* qui sont mis en *relation*. Les composants sont les suivants :

- L'utilisateur U
- Le système S comprend :
  - Les éléments du système, utiles à l'interaction en entrée S_{tool} ,

- Les éléments du système, utiles à la perception de l'état interne du système, $S_{representation}$. Ils sont décomposés eux-mêmes en :
  - L'objet de la tâche S_{object} ,
  - Les attributs de l'objet de la tâche S_{info} .

- Les objets réels R comprennent :
  - Les outils R_{tool} ,
  - Les objets de la tâche R_{object} .
- Les adaptateurs A transfèrent les données d'un monde à l'autre, et sont de deux types :
  - Les adaptateurs en entrée du système A_{in} ,
  - Les adaptateurs en sortie A_{out} .

Ces composants sont ensuite mis en relation. Une flèche (→) entre deux composants symbolise l'échange de données (physique ou numérique) entre les deux composants.
La Figure 35 présente une représentation de CASPER avec ASUR. Le chirurgien (composant U) manipule et voit l'aiguille de ponction (composant R_{tool}) qui est en contact physique avec le patient (composant R_{object}). La position de l'aiguille est capturée par un système de localisation optique (composant A_{in}). La position captée de l'aiguille constitue ensuite le composant S_{tool} qui est rendu observable via l'écran (composant A_{out}). La trajectoire de l'aiguille pré-calculée est elle aussi rendue observable via l'écran.

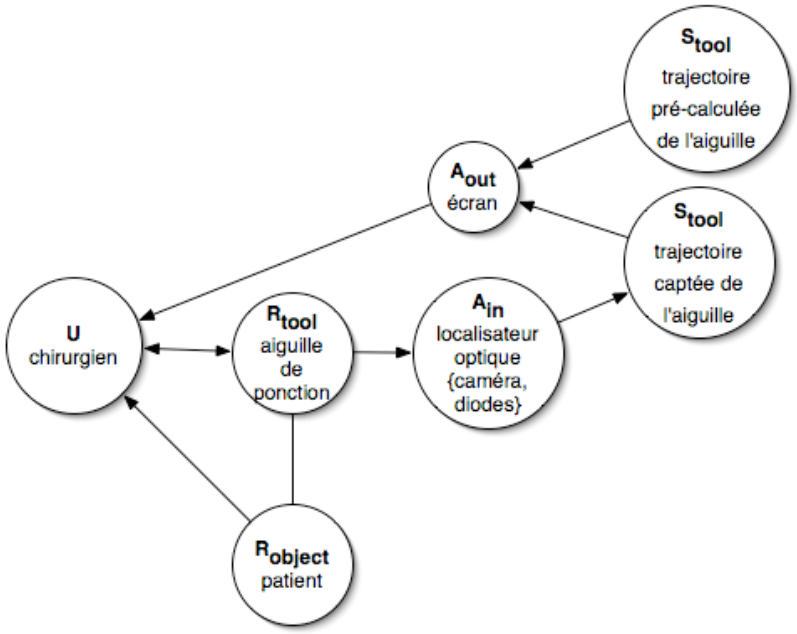


Figure 35 : Une modélisation de CASPER avec ASUR.

La notation ASUR s'accompagne de caractéristiques des éléments de la notation que nous présentons maintenant.

1.1.8.1. Caractéristiques des éléments de la notation

Des caractéristiques importantes pour la conception sont associées aux éléments de la notation ASUR (relations, composants, chemins dans un graphe).
Parmi les caractéristiques pour les relations qui décrivent les données échangées, nous trouvons :

- Le nombre de dimensions de la relation (1D, 2D, 3D, stéréoscopique, etc.),
- Le type de langage utilisé (texte, graphique, etc.),
- Le point de vue (égocentrique ou exocentrique).

Ces caractéristiques sont étendues dans [Dubois, Gray, 2007] :

- Le medium d'une relation est le moyen de transmettre l'information, par exemple la position de l'aiguille.
- La représentation est la forme de transmission de l'information par ce medium, qui pourrait être par exemple deux valeurs « bien placée »/« mal placée ». Cette représentation a plusieurs niveaux, comme une phrase en langage naturel, qui est une forme de représentation et qui peut elle-même être dite à l'orale ou écrite.

Parmi les caractéristiques pour les composants, nous trouvons :

- La position dans l'espace où le composant est perceptible et/ou modifiable,
- Le sens (canal sensoriel) ou l'action de l'utilisateur mis(e) en jeu pour percevoir ou modifier le composant. Cela peut être par exemple la vue ou le geste.

Ces caractéristiques sont étendues dans [Dubois, Gray, 2007] :

- La méthode de modification désigne la façon pour le composant d'affecter le medium, par exemple par l'action de l'utilisateur qui bouge l'aiguille de ponction.
- Le mécanisme de capture désigne les dispositifs et/ou processus qui permettent cette modification, par exemple des diodes et une caméra pour le localisateur optique.

Parmi les caractéristiques pour les chemins dans un graphe ASUR, nous trouvons :

- Le modèle de l'utilisateur voulu par le concepteur : par exemple bouger l'aiguille dans le contexte d'une salle d'opération est associé à la ponction péricardiale.
- La notion de groupe, qui est basée sur les chemins d'interaction dans un graphe ASUR. Cette notion permet de faire des regroupements logique dans le cas de graphes importants.

1.1.8.2. Réalité augmentée et virtualité augmentée

Les auteurs d'ASUR remarquent que la cible de l'interaction peut appartenir au monde physique ou au monde numérique. Ils reprennent alors le continuum de [Milgram, Kishino, 1994] pour le préciser. Ils distinguent la réalité augmentée de la virtualité augmentée :

- Dans la réalité augmentée, l'interaction avec le monde physique est augmentée avec l'informatique. C'est le cas de CASPER [Dubois et al., 1999] puisque la ponction péricardiale est une tâche qui se déroule dans le monde physique. C'est aussi le cas du système pour les contrôleurs aériens [Mackay, 1996] (section 2.1.2.1 du Chapitre 2) puisque leur tâche s'effectue dans le monde physique.
- Dans la virtualité augmentée, l'interaction avec le monde numérique est augmentée par des objets ou des actions physiques. C'est le cas par exemple de Carcade (section 2.2.1.1 du Chapitre 2) ou de Tuttuki Bako (section 2.2.2.2 du Chapitre 2), puisque les joueurs interagissent des objets numériques (un avion pour Carcade et un panda par exemple pour Tuttuki Bako).

1.1.8.3. Exécution et évaluation augmentée

En s'inspirant de la théorie de l'action de Norman [Norman, 1986], les auteurs distinguent l'exécution de l'évaluation augmentée :

- Un système de réalité mixte augmente l'exécution s'il réduit le gouffre de l'exécution [Norman, 1986] en ajoutant de nouvelles possibilités d'action ou en améliorant la qualité de la réalisation de la tâche. C'est le cas par exemple de Tuttuki Bako (section 2.2.2.2 du Chapitre 2) où l'exécution est augmentée en donnant l'illusion que l'on peut toucher directement avec le doigt le contenu numérique de la boîte.
- Un système de réalité mixte augmente l'évaluation s'il réduit le gouffre de l'évaluation [Norman, 1986] en ajoutant la possibilité de percevoir des données qui ne serait pas sans le système. C'est l'exemple de CASPER, où le chirurgien peut percevoir sur l'écran la déviation de son aiguille de ponction par rapport à la trajectoire idéale calculée.

1.1.8.4. Continuité et compatibilité perceptuelles et cognitives

En adaptant les propriétés ergonomiques d'observabilité et d'honnêteté [Abowd et al., 1992], les continuités et compatibilités perceptuelles et cognitives ont aussi été identifiées [Dubois, 2001]. L'observabilité caractérise la capacité du système à permettre à l'utilisateur de percevoir l'état interne du système. L'honnêteté caractérise la capacité du système à permettre à l'utilisateur une interprétation correcte de l'état interne du système. Ces propriétés ont été étendues pour le cas de plusieurs concepts :

- La compatibilité perceptuelle est l'extension de l'observabilité au cas de plusieurs concepts. C'est la possibilité pour l'utilisateur de percevoir les données relatives aux différents concepts en prenant en compte leur dispersion dans l'espace et les sens perceptifs mis en jeu. La compatibilité perceptuelle implique l'observabilité de chacun des concepts considérés isolément. Par exemple dans CASPER, les positions captée et idéale de l'aiguille sont compatibles perceptuellement puisque affichées sur le même écran (Figure 36).

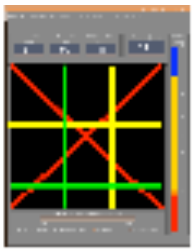


Figure 36 : Le système CASPER et la visualisation des trajectoires de l'aiguille de ponction sur l'écran (d'après [Dubois, 2001]).

- La compatibilité cognitive est l'extension de l'honnêteté au cas de plusieurs concepts. C'est la possibilité pour l'utilisateur d'interpréter correctement les données relatives aux différents concepts en prenant en compte leurs différents modes de représentation. La compatibilité cognitive implique l'honnêteté des représentations de chacun des concepts considérés isolément. Par exemple dans CASPER, les positions captée et idéale de l'aiguille sont compatibles cognitivement puisque affichées de la même façon, sous forme de croix (Figure 36).

L'observabilité et l'honnêteté ont été également étendues au cas d'un unique concept qui est représenté plusieurs fois (représentation multiple d'un même concept, [Abowd et al., 1992]).

- La continuité perceptuelle est la possibilité pour l'utilisateur de percevoir les données relatives au concept en prenant en compte leur dispersion dans l'espace et les sens perceptifs mis en jeu. Par exemple dans SandScape (présenté à la section 2.1.1.2 du Chapitre 2), il y a une discontinuité perceptuelle puisque la simulation est représentée superposée au sable sur la table (Figure 37) et aussi sur le mur d'en face (Figure 37).

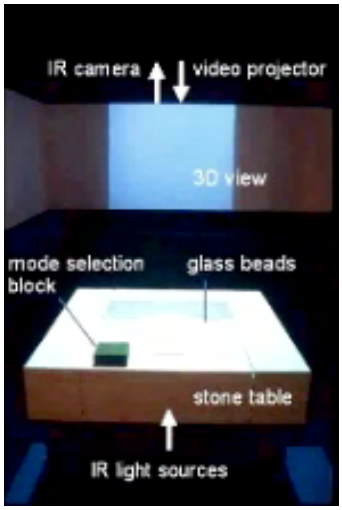


Figure 37 : L'installation matérielle du système SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg.

- La continuité cognitive est la possibilité pour l'utilisateur d'interpréter correctement les données relatives au concept en prenant en compte ses différents modes de représentation. Par exemple dans SandScape, une des representations est la projection d'une image en 2D (sur la table, Figure 13) et l'autre est la projection d'une image en 3D (sur le mur, Figure 13).

En résumé, ASUR est une notation de conception, associée à des caractéristiques. Cet ensemble permet de décrire le couplage entre mondes physique et numérique. ASUR traite donc l'enjeu des interfaces mixtes lié à la multiplication des possibilités de couplage entre physique et numérique (Chapitre 2, section 3.1.1). ASUR traite aussi explicitement le thème de l'utilisation de l'espace (Chapitre 2, section 3.1.3) grâce à ses caractéristiques. De plus, par l'articulation de la notation de conception avec des critères ergonomiques [Charfi et al., 2008], ASUR fourni au concepteur des recommandations ergonomiques utilisables en conception, afin de favoriser l'utilisabilité de l'interfaces de réalité mixte conçue.

1.1.9. IRVO

Le modèle IRVO (*Interacting with Real and Virtual Objects*[18]) est aussi une notation dédiée à la réalité mixte. IRVO [Chalon, 2004] comprend plusieurs catégories d'entités, illustrées à la Figure 38 sur l'exemple de CASPER :

- L'utilisateur U, représenté avec ses canaux sensoriels d'entrée/sortie
  - KH pour le toucher, le geste ;
  - V pour la vue ;
  - A pour l'ouïe et la voix ;
- Les objets sont de deux types :
  - L'objet de la tâche O, qui peut être
    - Purement physique Or,
    - Physique et augmenté Or+v (cas de la réalité augmentée, définie par [Dubois, 2001] et présentée à la section 1.1.8.2 de ce chapitre),
    - Purement numérique Ov ,
    - Numérique et augmenté Ov+r (cas de la virtualité augmentée, définie par [Dubois, 2001] et présentée à la section 1.1.8.2 de ce chapitre),
  - Les outils T, qui peuvent se décliner comme les objets de la tâche en Tr, Tr+v , Tv , Tv+r .
- Les transducteurs sont de type senseur S ou effecteur E.

[18] Interagir avec des objets réels et virtuels

- Le modèle interne de l'application M, c'est-à-dire le système informatique sans les objets, qui peuvent être mixtes ou numériques. Il n'est pas forcément présent dans un schéma IRVO, puisqu'il n'interagit pas directement avec l'utilisateur.

Dans l'exemple de CASPER dans sa version avec casque semi-transparent, présenté avec IRVO à la Figure 38, l'utilisateur manipule (canal KH) l'aiguille et son support (Tr). Sa position est captée par un transducteur (S) qui la localise et transmet cette information à la partie numérique de l'aiguille (T+v). Cette partie numérique est rendue perceptible via le casque (effecteur E). L'aiguille et son support, ainsi que l'image de la partie numérique dans le casque sont vues (canal V) superposées (+). De l'autre côté, le patient (Or) et la trajectoire idéale dans la région péricardiale (O+v) rendue perceptible via le casque (E) sont vus par l'utilisateur. Entre ces deux ensembles (T et O), l'aiguille (T) agit sur le liquide péricardial du patient (O).

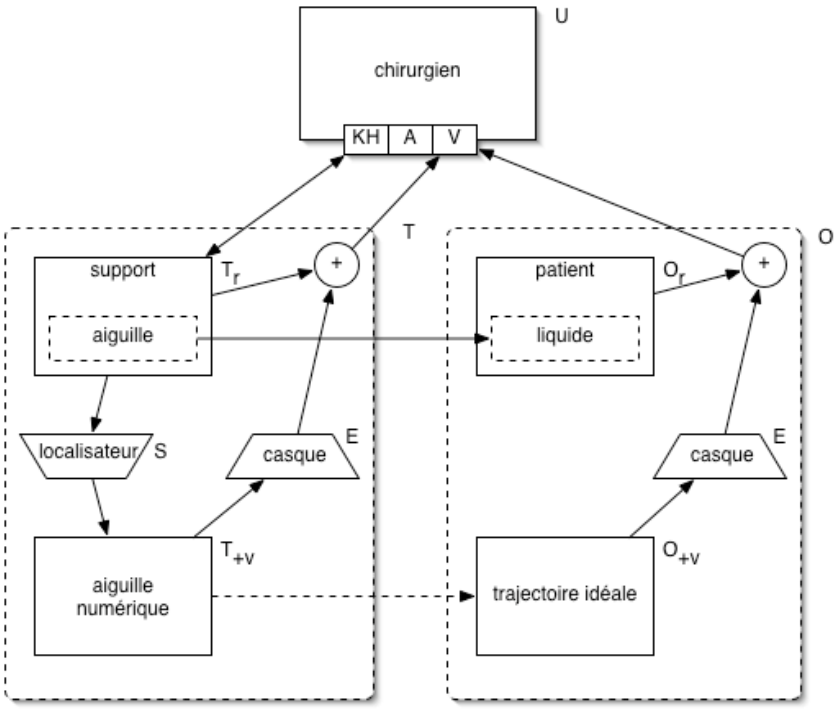


Figure 38: Modélisation de CASPER avec IRVO.

Plusieurs concepts sont en commun avec ASUR, comme

- La notion d'outil et d'objet de la tâche dans l'interaction avec les éléments T (*tool*[19]) et O (*object*[20]) pour IRVO et, pour ASUR, les éléments S_{tool} et R_{tool} d'un côté pour les outils, et S_{object} et R_{object} de l'autre côté pour les objets de la tâche,
- La réalité augmentée et la virtualité augmentée, avec les éléments r+v (cas de la réalité augmentée, définie par [Dubois, 2001] et présentée à la section 1.1.8.2 de ce chapitre) et v+r (cas de la virtualité augmentée, définie par [Dubois, 2001] et présentée à la section 1.1.8.2 de ce chapitre),
- Les transducteurs qui correspondent aux adaptateurs d'ASUR.

Néanmoins, tandis qu'IRVO associe les canaux sensoriels humains au niveau du composant Utilisateur, ASUR les décrit au niveau des échanges (flèches) qui sont liées à l'utilisateur.

Nous soulignons que, parmi les enjeux de recherche liés aux interfaces mixtes présentés au Chapitre 2, la notation de conception IRVO considère notamment les enjeux de conception liés à la multiplication des possibilités de couplage entre physique et numérique (Chapitre 2, section 3.1.1).

Nous venons de présenter différents travaux généraux sur les systèmes de réalité mixte. Nous présentons maintenant des outils conceptuels qui se réclament des interfaces tangibles.

[19] outil
[20] objet

1.2. Outils conceptuels pour les interfaces tangibles

Nous avons défini les interfaces tangibles à la section 1 du Chapitre 2 comme des interfaces où les objets mis en jeu dans l'interaction sont tangibles, palpables. Les outils conceptuels qui se concentrent sur les interfaces tangibles étudient la caractérisation des objets manipulés par l'utilisateur du système. Nous les présentons maintenant.

1.2.1. MCRit

MCRit [Ullmer et al., 2005] est modèle d'interaction pour les interfaces tangibles, qui s'inspire de l'architecture logicielle MVC (Modèle Vue Contrôleur) introduit pour les interfaces graphiques. MCRit est l'abréviation de Modèle Contrôle Représentation, intangible et tangible.

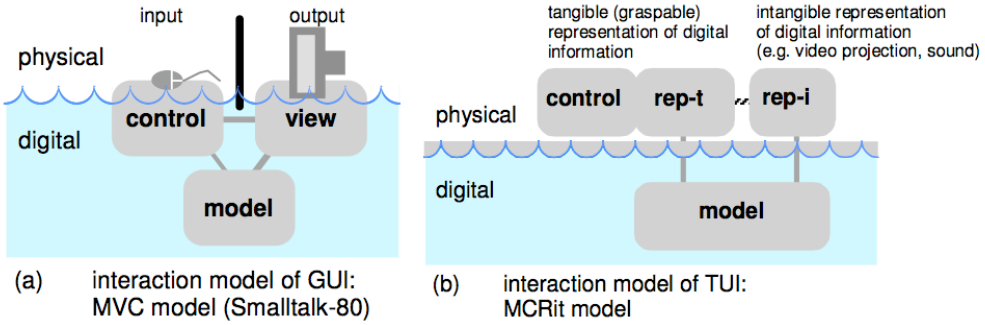


Figure 39: Modèle d'architecture logicielle MVC (a) et le modèle d'interaction MCRit (b). Illustration extraite de [Ullmer et al., 2005].

Dans ce modèle d'interaction, le composant Vue de MVC est décomposé en deux représentations : tangible (manipulable) et intangible (comme les représentations graphiques ou le son). La partie tangible des représentations (sortie du système) sert aussi de contrôleur (entrée du système). Par exemple dans le DigitalDesk, la gomme tangible représente la fonction associée à l'objet, mais est aussi utilisée comme dispositif d'interaction en entrée du système : c'est de la gomme que se saisit l'utilisatrice pour gommer son dessin. Puisqu'il prend en compte des parties tangibles, qui ne sont pas logicielles, MCRit n'est donc plus une architecture logicielle mais une description de l'interaction qui s'inspire d'un modèle d'architecture logicielle.

1.2.2. Continuum des objets

Dans [Underkoffler, Ishii, 1999], les auteurs présentent une taxonomie des objets utilisés dans les interfaces tangibles. Les auteurs distinguent cinq types d'objets qui participent à l'interaction :

- Les objets « purs » : Ce sont des objets qui interviennent dans l'interaction sans que leurs caractéristiques soient importantes. Ce pourrait être l'exemple d'une interaction avec le DigitalDesk où il suffirait de poser un objet sur la table pour démarrer le système. Peu importe les caractéristiques de cet objet, il suffit qu'il soit présent.
- Les objets « attributs » : Ce sont les objets dont le système interactif ne considère qu'un unique attribut et fait abstraction des autres. Ce peut être l'exemple des objets utilisés dans [Carvey et al., 2006] où seul le poids d'un objet est pris en compte. Nous attirons l'attention sur le fait que si le système informatique ne prend en compte qu'un attribut de cet objet, l'utilisateur lui peut prendre en compte d'autres attributs supplémentaires, comme un porte-clés associé à un mot de passe via une balance USB. L'utilisateur peut considérer dans ce cas la métaphore associée. Nous remarquons que ces attributs peuvent entrer en conflit : si l'utilisateur rajoute une clé à porte-clés, le lien ne fonctionne plus, au contraire de la métaphore.
- Les objets « noms » : Ce sont les objets de la tâche, comme le dessin de la maison dans le DigitalDesk.

- Les objets « verbes » : Ce sont les outils, qui permettent d'agir sur les autres. C'est par exemple la gomme ou le bouton *FILL* de papier dans le DigitalDesk. On retrouve ici la notion d'outil présente dans ASUR et IRVO.
- Les objets « outils reconfigurables » : Ce sont des objets « verbes » dont l'utilisateur peut changer la fonction. Ce pourrait être un bouton de papier comme celui du DigitalDesk, mais où l'utilisateur pourrait remplacer la commande *FILL* par une autre effaçant et réécrivant par-dessus.

1.2.3. Outil, objet, conteneur

Dans [Holmquist et al., 1999] est présentée une autre taxonomie des objets utilisés dans les interfaces tangibles. Les auteurs distinguent les outils qui servent à manipuler l'information, les objets qui représentent et mémorisent de l'information, et les conteneurs qui sont des outils permettent de déplacer de l'information d'un dispositif ou d'une plateforme à l'autre.
Nous généralisons cette approche en considérant (Tableau 2) qu'un outil et un objet peuvent être génériques (conteneurs) ou spécifiques. Par exemple dans CASPER, l'aiguille est un outil spécifique, alors que dans [Carvey et al., 2006] (section 1 du Chapitre 2) les objets associés à une fonctionnalité numérique via la balance USB sont des outils génériques, car ils peuvent être associés à n'importe quelle fonctionnalité. De même un objet comme le patient de CASPER est un objet spécifique, alors que les *Voice Boxes* [Jeremijenko, 1995] (présenté à la section 2.3.3 du Chapitre 2) sont des objets génériques qui peuvent contenir d'importe quel son.

	Outil	**Objet**
Générique	Outil générique ou conteneur d'outil	Objet générique ou conteneur d'objet
Spécifique	Outil	Objet

Tableau 2 : Généralisation de la taxonomie outil, objet, conteneur de [Holmquist et al., 1999].

1.2.4. Taxonomie des interfaces tangibles

Dans [Fishkin, 2004], nous trouvons une classification des interfaces tangibles selon deux axes : les métaphores et l'incarnation.

1. L'auteur distingue deux types de métaphores :
   a. Une **métaphore de nom** est définie comme « un <x> dans le système est comme un <x> dans le monde réel ». Par exemple, dans le DigitalDesk, une gomme est comme une gomme dans le monde réel, elle permet de gommer. À cette étape, la gomme n'est pas nécessairement utilisée de la même façon que dans le monde réel : cette caractéristique concerne le second type de métaphore.
   b. Une **métaphore de verbe** est définie comme « faire une action <x> dans le système est comme faire la même action dans le monde réel ». Avec le même exemple du DigitalDesk, gommer avec le système se fait de la même façon que gommer dans le monde réel. Cette métaphore pourrait être appliquée même si l'outil n'avait pas la forme d'une gomme, avec un objet qu'on frotterait contre le papier pour gommer.

   Les métaphores définissent quatre classes d'interfaces tangibles : Une interface peut ne présenter aucune métaphore, présenter une métaphore de nom ou une métaphore verbe, ou enfin présenter une métaphore à la fois de nom et de verbe. L'exemple de la gomme présente les deux métaphores simultanément. Nous soulignons que nous retrouvons ces concepts (nom et verbe) dans l'approche présentée plus haut (Continuum des objets), ici appliqués au cas de la métaphore.

2. Le second axe de la taxonomie est l'incarnation[21]. L'incarnation est définie par la relation spatiale entre l'entrée et la sortie correspondant au même concept : Il peut être complet, proche, environnemental ou distant. Par exemple, dans SandScape [Ishii et al., 2004], les

[21] *embodiment*

utilisateurs manipulent du sable sur une table pour comprendre le relief à travers plusieurs simulations (altitude, ensoleillement, écoulement, etc.). La représentation de la simulation sur le sable est à la fois complète, puisque le sable est l'entrée (dispositif d'interaction associé à une caméra) et la sortie (le support de projection), mais aussi distante, puisqu'une vue en trois dimensions de la même simulation est projetée sur le mur. Cet axe fait écho à la continuité perceptuelle de [Dubois, 2001], présenté plus haut à la section 1.1.8.4.

1.2.5. Taxonomie des interfaces saisissables

Une taxonomie des interfaces tangibles est présentée dans [Fitzmaurice et al., 1995]. Les auteurs nomment « briques » les objets tangibles manipulés l'utilisateur. La taxonomie proposée caractérise les interfaces tangibles incluant des briques. Nous retenons les axes suivants :

1. **Capacité interne de l'objet** : Inerte, simple, ou intelligent.
2. **Entrée et Sortie** : liste des propriétés captées et des propriétés observables.
3. **Conscience spatiale** : Un objet peut être isolé, connaître la position des autres briques, et/dou avoir des informations sur son environnement.
4. **Connexion :** sans fil, câblée, ou sur une grille.
5. **Durée de l'interaction** : une interaction élémentaire avec une brique peut durer de quelques secondes (un geste) à plusieurs heures (voire plus). Ce serait l'exemple d'une brique qui déclenche un processus seulement si l'utilisateur ne la déplace pas pendant plusieurs mois.
6. **Nombre de briques utilisées en parallèle**.
7. **Assignation des fonctions** : permanent, programmable, re-assignation rapide. Cet axe fait écho au continuum des objets (objets verbes et outils reconfigurables) [Underkoffler, Ishii, 1999] et aux outils/objets/conteneurs [Holmquist et al., 1999] qui viennent d'être présentés respectivement en sections 2 et 3.
8. **Représentations de l'interaction** : Quelle est la part des objets numériques et des objets physiques parmi les objets disponibles pour l'interaction ?
9. **Composition spatiale entre les éléments physiques** (briques) **et numériques** (par exemple projection) : superposés ou non.
10. **Temporalité du lien** entre physique et numérique : La brique est liée vers le monde numérique en temps réel ou en temps différé. Nous soulignons que ce concept complète la temporalité du lien (éphémère/persistant) caractérisée dans [Renevier, 2004].
11. **Granularité** (précision spatiale) : table, pièce, bâtiment. Par exemple si la brique connaît sa position, elle peut la connaître au millimètre près sur la table, plus grossièrement dans la pièce, ou même connaître dans quelle pièce ou étage du bâtiment elle est.
12. **Dynamicité de la surface d'opération** : La brique est utilisée sur une surface statique ou dynamique (écran).
13. **Texture de la surface d'opération** : discrète (la brique est positionnée sur une grille, comme l'axe connexion (4)) ou continue (la brique peut être déplacée continûment sur une surface plane).

Cette taxonomie aborde les enjeux de conception présentés au Chapitre 2 liées à l'utilisation de l'espace (Chapitre 2, section 3.1.3), à la multiplication des possibilités de liens entre physique et numérique (Chapitre 2, section 3.1.1), et à la prise en compte du contexte dans l'interaction (Chapitre 2, section 3.2.1). En revanche, certains de ces axes ne s'appliquent qu'à un cas particulier d'interfaces, avec des briques sur une surface.

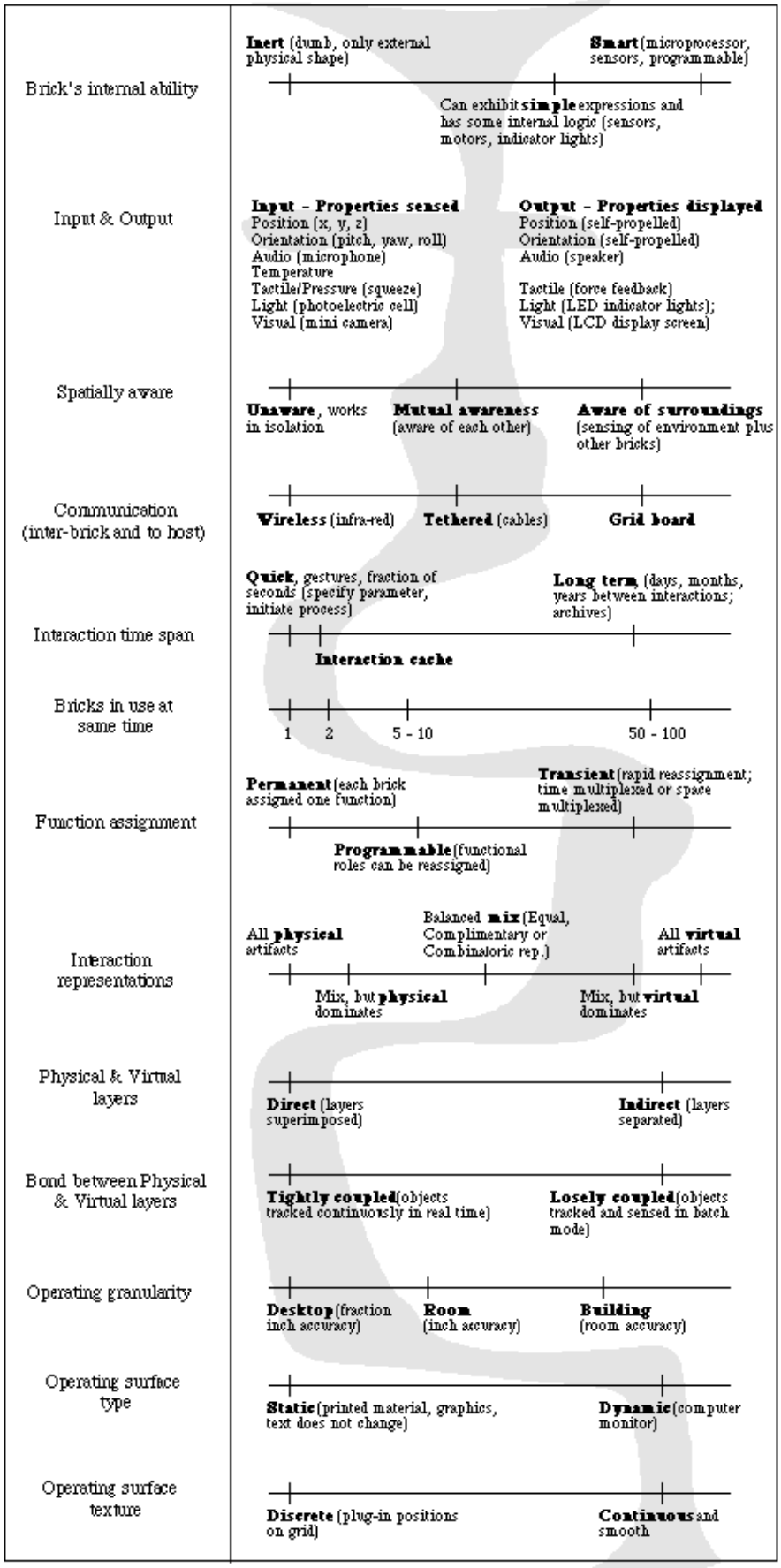


Figure 40: Taxonomie des interfaces tangibles. Illustration extraite de [Fitzmaurice et al., 1995].

1.2.6. Contraintes

La notion de contrainte dans l'interaction est souvent étudiée dans les interfaces tangibles [Shaer et al., 2004.] [Ullmer et al., 2005] [Patten et al., 2007].
TAC (sigle de *Token And Constraints*[22]) est une description de l'interaction proposée dans [Shaer et al., 2004.]. TAC met en avant les relations entre objets physiques et informations numériques via des entités de types suivants :

- **Pyfo** : Un « pyfo » est un objet physique. Lui-même peut être composé de plusieurs pyfos, comme un cube est composé de 6 faces. Les auteurs ont choisi la terminologie « pyfo » à la place d'objet, trouvant ce dernier mot déjà chargé de sens en informatique. Les auteurs ont également voulu éviter « objet physique », pour choisir un mot spécifique pour les objets participants aux interfaces. Les *pyfos* sont de 2 types : les tokens et les contraintes. Un pyfo peut être l'un ou l'autre, de façon non exclusive.
- **Token** : Un token est un objet physique qui représente une information numérique (c'est-à-dire la variable présentée plus bas). L'utilisateur peut accéder à ou manipuler cette information en interagissant avec le token. Les auteurs remarquent qu'à ce token peut être appliqué une métaphore, comme dans [Fishkin, 2004] et que sa forme peut etre étudiée à la lumière de l'affordance [Norman].
- **Contrainte** : Une contrainte est un objet physique associé à un pyfo et qui limite ce que l'utilisateur peut faire avec le pyfo auquel il est associé. Cette contrainte peut (1) suggérer à l'utilisateur comment manipuler (ou comment ne pas manipuler) le token associé. Il peut aussi (2) limiter l'espace d'interaction ou (3) servir de référentiel pour interpréter une relation TAC (*Token And Contraint*). On retrouve ici les concepts de mouvements captés, attendus et souhaités de [Benford et al., 2005] (section 1.1.7 de ce chapitre). Par exemple dans SandScape, la table sur laquelle les utilisateurs interagissent avec le système sert à la fois à limiter l'espace d'interaction avec le sable, à suggérer où poser le marqueur tangible, grâce aux niches. Enfin la table peut aussi être considérée comme référentiel pour le marqueur tangible, puisque la position du marqueur est interprétée relativement à la table, et non pas de façon absolue.
- **Variable** : Une variable est une entité qui représente de l'information numérique. Elle peut être (ou ne pas être) associée à un token.
- **Relation TAC** : une relation entre un token, une variable, et une (ou plusieurs) contrainte(s). Cette relation peut être temporaire ou non. Nous soulignons ici que cette caractéristique de la relation TAC rejoint la caractéristique temporelle des liens (éphémère/persistant) de [Renevier, 2004] présenté à la section 1.1.5 de ce chapitre. La possibilité d'une relation est prévue par un concepteur, mais c'est l'utilisateur ou le concepteur qui peut mettre ces objets en relations. Lorsque l'utilisateur manipule une relation TAC, il manipule le token en respectant ses contraintes associées. Cette manipulation a des conséquences sur la variable associée dans le monde numérique.

Le paradigme TAC est constitué de cinq propriétés :

- **Couple** : Un pyfo couplé à une variable est un token. Le concepteur décide quel type de variable peut être associé à un pyfo particulier. Le couplage effectif peut quant à lui apparaître en conception ou pendant l'utilisation : Il s'agit de lien statique ou dynamique, comme identifié dans [Renevier, 2004].
- **Définition relative** : Un pyfo peut être un token ou une contrainte, ou les deux.
- **Association** : Une relation TAC est créée quand un token est associé à une contrainte. De nouvelles contraintes peuvent ensuite s'y ajouter.
- **Interprétation informatique** : La manipulation d'une relation TAC est interprétée par le système. Si le token de ce TAC est manipulé en dehors de ses contraintes, alors le système n'en tient pas compte. Nous soulignons ici que ce concept est présent dans [Benford et al., 2005] avec les mouvements attendus, captés et souhaités (section 1.1.7 de ce chapitre).

[22] Jetons/objets et contraintes

- **Manipulation** : Un TAC peut être manipulé de façon discrète, continue, ou les deux successivement. Ce sont les contraintes de ce TAC qui suggèrent comment manipuler le TAC.

Il est possible d'utiliser ces résultats pour spécifier une interface. Le Tableau 3 montre les spécifications de l'interaction avec la gomme du DigitalDesk dans les termes de TAC.

TAC	Représentation		Comportement		
	Token	*Contrainte*	*Variable*	*Action*	*Retour d'information*
1	Gomme	Dessin	Gommer	Bouger comme une gomme	
2	Dessin	Table, Gomme	Dessin numérique	Dessiner dessus Utiliser un autre token dessus (Bouton de papier *FILL*, Gomme)	Projection du dessin numérique, superposée au dessin

Tableau 3: Spécification de l'interaction avec la gomme du DigitalDesk dans les termes de TAC.

Dans [Ullmer et al., 2005] et [Ullmer, Ishii, 2001], le rôle des contraintes dans l'interaction est également étudié. Dans [Ullmer, Ishii, 2001], les auteurs introduisent notamment les notions d'interaction spatiale, relationnelle, et constructive. Ce sont des types d'association entre la répartition des objets dans l'espace et l'interprétation qui en est faite par le système.
Si l'interaction est **spatiale**, c'est uniquement la position absolue des objets qui est prise en compte par le système. C'est l'exemple des reliefs de sable dans SandScape.
Dans le cas d'une interaction **relationnelle**, c'est la position relative aux autres objets qui participent à l'interaction et est interprétée par le système. C'est le cas du DigitalDesk par exemple où la relation entre le bouton de papier *FILL* et le dessin est interprétée.
Dans le cas d'une interaction **constructive**, les objets-tokens sont considérés comme des modules qu'il faut assembler pour former un nouvel objet qui sera interprété par le système.

La notion de relation TAC est affinée dans [Patten et al., 2007]. Les auteurs ajoutent la possibilité pour l'utilisateur d'associer lui-même un token avec une contrainte. Ces cas de mise en relation spontanée ne sont pas prévus par le concepteur, et ne sont donc par conséquent pas interprétés par le système, comme dans PICO [Patten et al., 2007] (section 2.1.1.3 du Chapitre 2). Dans cet exemple, l'utilisateur pose un objet rempli de sable sur un autre pour l'empêcher de se déplacer, et c'est uniquement la position du token qui est interprétée par le système. Dans [Patten et al., 2007], l'utilisateur lui-même peut donc concevoir des relations TAC.

1.2.7. Physicalité

[Dix et al., 2007] proposent un modèle pour décrire des interfaces mixtes, également valable pour les dispositifs physiques ou l'informatique/électronique embarquée « traditionnelle », comme une machine à laver. Ce modèle est basé sur une description des entrées/sorties du système que nous présentons d'abord. Nous présentons ensuite le modèle formel, basé sur cette description et sur les automates à état.

1.2.7.1. Description des entrées/sorties du système

Dans la description des entrées/sorties du système présenté Figure 41, l'interaction est décrite de la façon suivante : L'utilisateur manipule l'objet physique (a), et perçoit la première conséquence de son action physique via la boucle d'interaction (A) : il constate l'état de l'objet physique (b) et/ou perçoit les effets de son action (c). Par exemple, si l'utilisateur tourne un bouton, il voit le changement d'orientation du bouton (b) et perçoit la résistance du bouton sous ses doigts (c).

Pour interagir avec le système, cette action (a) doit être captée (i). Afin de signifier à l'utilisateur que son action a été captée, le système peut mettre en route un effecteur (ii) via la boucle d'interaction (B). Du point de vue de l'utilisateur, ceci donnera lieu à un retour d'information parfois indissociable des premiers retours physiques (b) et (c). Par exemple, si l'utilisateur tourne un bouton (a), il peut voir une diode s'allumer (b) aussi longtemps qu'il tourne le bouton.
De plus, les actions captées (i) peuvent changer l'état interne du système via la boucle d'interaction (C). Ce changement dans l'état interne du système est rendu observable via un retour du système vers l'utilisateur de (iii) vers (d). Par exemple, si l'utilisateur tourne le bouton (a) pour monter le volume, il peut voir afficher sur un écran « volume = 23 » (d).
Enfin, la dernier boucle (D) est la boucle d'interaction sémantique. La finalité de l'action peut être de changer l'état d'un objet physique. Ce changement d'état est perceptible via les retours d'information (iv) et (e). Par exemple, si l'utilisateur tourne le bouton pour monter le volume, il entend effectivement la musique plus fort.
Nous soulignons que cette description de l'interaction présente des concepts communs avec le modèle d'interaction instrumental [Beaudouin-Lafon, 2004] : l'instrument sert de médiateur, en entrée et en sortie, entre l'utilisateur et le système.

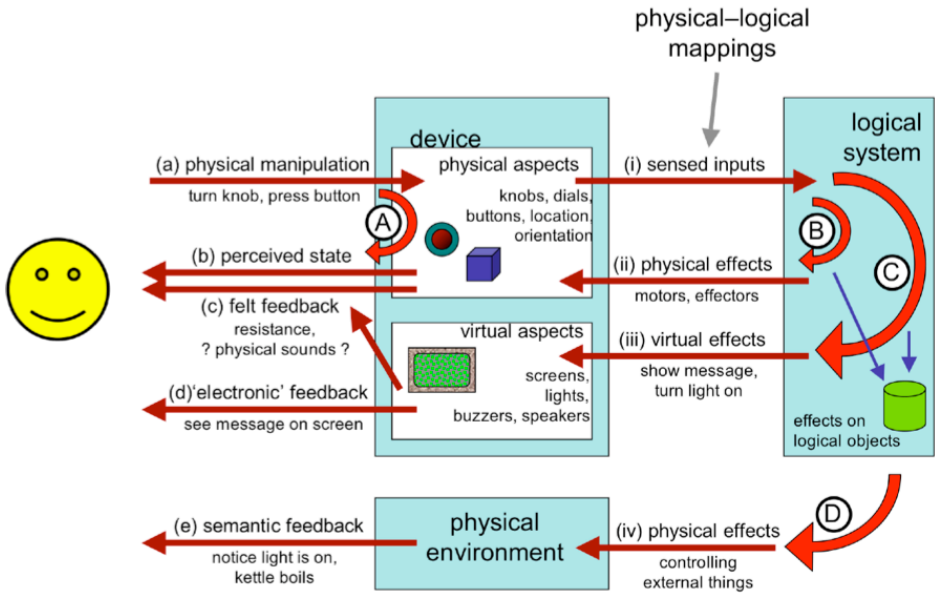


Figure 41: Différents niveaux de retour d'information. Illustration extraite de [Dix et al., 2007].

Nous présentons maintenant le modèle formel, basé sur les automates à état et sur la description des entrées/sorties du système que nous venons de présenter.

1.2.7.2. Modèle formel

Le modèle de [Dix et al., 2007] adopte une approche basée sur les automates à état pour décrire l'interaction, permettant de mettre en relation l'état de la partie physique du système et la partie logique. Par exemple, un interrupteur pour allumer la lumière dans une pièce est décrit à la Figure 42. Dans cet exemple, il n'y a pas de correspondance directe entre l'état physique (*up* ou *down*), et l'état logique (lumière allumée ou éteinte), mais un changement d'état physique implique un changement d'état logique.

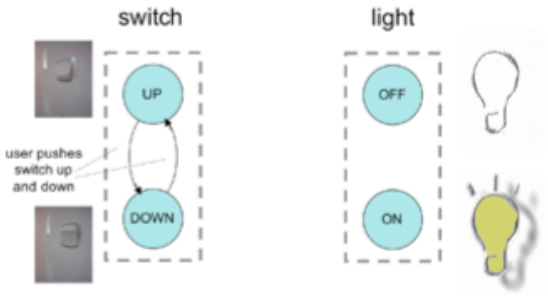


Figure 42: Interrupteur pour allumer la lumière. Illustration extraite de [Dix et al., 2007].

Nous présentons d'abord le cas général (i) du modèle formel, puis ses trois extensions dans le cas des boutons élastiques ou à rebond (ii), dans le cas des transitions dépendantes du temps (iii), et dans le cas des transitions initiées par le système (iv).

1.2.7.2.1. Cas général

UA l'ensemble des actions potentielles des utilisateurs (*User Actions*). *PA* désigne l'ensemble de ce qui est perceptible (*Perceivable Attributes*). Un dispositif physique est décrit comme un automate avec des états et des transitions, avec :

- *DS*, l'ensemble des états physiques du dispositif (*Device's State*)
- $DT \subseteq DS \times DS$, l'ensemble des transitions possibles entre les états. Certaines transitions directes entre deux états physiques peuvent être impossibles. Les transitions possibles sont donc un sous-ensemble de *DT*.

Certaines de ces transitions sont contrôlées par l'utilisateur : $action : UA \leftrightarrow DT$
Par cette relation nommée *action*, un élément de *UA* peut avoir plusieurs images dans *DT* (plusieurs éléments de DT peuvent correspondre à un élément de UA par la relation *action*), et un élément de *DT* peut avoir plusieurs antécédents dans *UA*. En effet, la même action de l'utilisateur peut avoir différents effets selon l'état du système, et une transition peut être causée par différentes actions de l'utilisateur (par exemple appuyer avec la main gauche ou la main droite).

Du côté logique (Figure 41), *LS* est l'ensemble des états logiques du système (*Logical States*). Pour chaque système, il est possible de définir quels états physiques et logiques pouvant être vérifiés simultanément, d'où la relation : $state\text{-}mapping : DS \leftrightarrow LS$

Nous présentons maintenant ce cas général étendu au cas des boutons élastiques ou à rebond.

1.2.7.2.2. Cas des boutons élastiques ou à rebond

Il existe des dispositifs physiques où un état peut être instable, à l'instar du bouton de la souris ou des boutons pour allumer un ordinateur. Une fois pressé, il revient vers son état initial sans intervention de l'utilisateur. Dans ce cas, la boucle de retour d'information A (Figure 41) n'existe pas et l'état du système peut alors ne pas être perceptible : c'est ce que les auteurs appellent un « état caché » (*hidden state*) [Dix et al., 2007].
Pour ces dispositifs, les auteurs ont ajouté au modèle formel précédent un nouveau type de transition : $Z : UA \leftrightarrow DT$
Les états qui peuvent être quittés par une transition de type *Z* sont appelés transitoires :
$\forall\ a \in UA, transitory\text{-}state(a) \Leftrightarrow \{ s \in DS\ tels\ que\ \exists (s, s') \in Z(a) \}$
Ces états éphémères sont représentés graphiquement dans l'automate de la Figure 43 par un contour pointillé et le nouveau type de transitions associé par une ligne en forme de ressort.

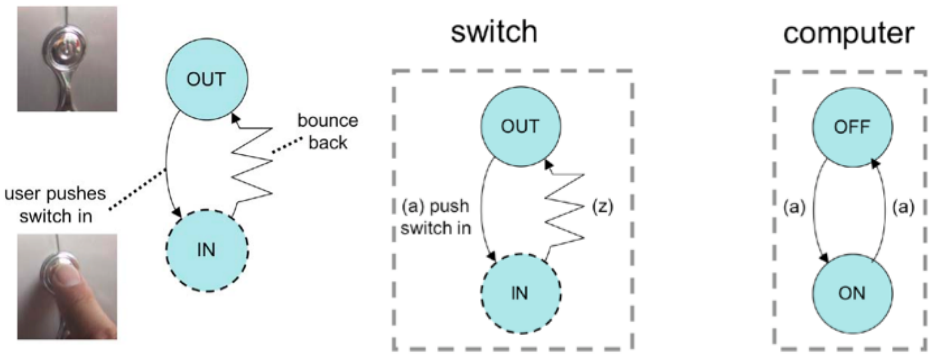


Figure 43 : Bouton élastique ou à rebond contrôlant l'alimentation d'un ordinateur. Illustration extraite de [Dix et al., 2007].

Les évènements *Ev*, dont (a) à la Figure 43 fait partie, entraînent des transitions dans la partie logique du système, d'où la définition de la relation *doit* : $Ev \times LS \rightarrow LS$

Les auteurs remarquent aussi que certains boutons, comme les interrupteurs de la Figure 44, reviennent à leur état initial jusqu'à ce que l'utilisateur pressent suffisamment fort pour qu'ils basculent dans un autre état physique. Le seuil de basculement est représenté par les états en pointillé *PART UP* et *PART DOWN* à la Figure 44 (gauche). Les auteurs représentent cette interaction dans l'automate avec les états en pointillé et les transitions de type « éclair » de la Figure 44 (gauche). Ils simplifient cette notation avec le schéma de la Figure 44 (droite) en réunissant les états avec les états éphémères correspondants. L'état *UP* avec les deux accolades de droite correspond aux états *UP* et *PART DOWN* de gauche, et l'état *DOWN* avec les deux accolades de droite *DOWN* et *PART UP* de gauche. Ainsi l'état *UP* avec les deux accolades correspond à l'état *UP* avant basculement vers le bas, et l'état *DOWN* avec les deux accolades correspond à l'état *DOWN* avant basculement vers le haut.

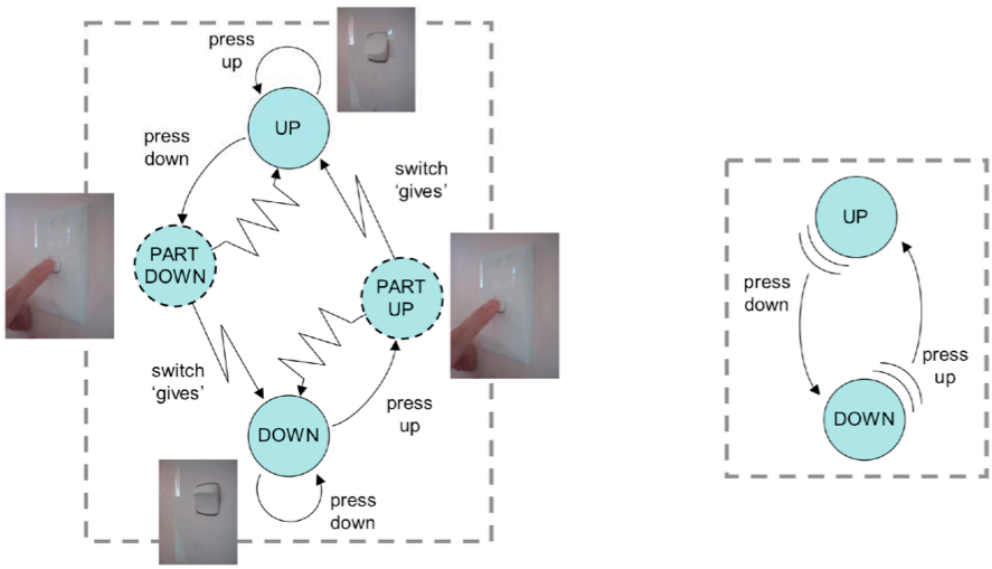


Figure 44 : Interrupteur pour lequel l'utilisateur doit presser suffisamment fort pour qu'il basculent dans un autre état physique. Illustration extraite de [Dix et al., 2007].

Nous présentons maintenant le cas étendu des transitions dépendantes du temps.

1.2.7.2.3. Cas des transitions dépendantes du temps

Certaines interfaces déclenchent des transitions selon le temps passé dans un état physique. Par exemple si on presse rapidement une touche du clavier d'un ordinateur, un seul évènement se produit. Au contraire si on laisse longtemps le doigt appuyé sur la même touche, alors l'évènement est répété à intervalle régulier aussi longtemps que le doigt est appuyé.

Si un état de DS est dépendant du temps, alors on a besoin de savoir de quelle façon l'interface est manipulée : on note *Time* la durée, et *Kind* un type qui peut prendre 2 valeurs : périodique ou non. Une transition dépendante du temps est alors définie par la relation :
time-trigger: $DS \times Time \times Kind \rightarrow Ev$
Un tel état est un état de tension, qui revient à son état d'équilibre s'il est relâché.

Nous présentons maintenant le cas étendu des transitions initiées par le système.

1.2.7.2.4. Cas des transitions initiées par le système

Un changement d'état physique peut être déclenché par le système. Les auteurs considèrent par exemple le bouton d'une bouilloire électrique. Un autre exemple est le déplacement des jetons par le système PICO [Patten et al., 2007]. Pour décrire ces transitions, un nouvel ensemble d'actions système (*SA*) et une nouvelle relation entre les états logiques et les actions système sont définis :
sys-trigger : LS → set(SA)
Ces actions système ont des effets sur les transitions, comme les actions de l'utilisateur :
sys-action : SA ↔ DT
Si un état logique (s) du système peut déclencher une action (a) de la part du système, alors au moins un état physique (d) correspondant à cet état logique (s) prend en compte cette action système (a) :
$\forall a \in SA, s \in LS, a \in \text{sys-trigger}(s) \Rightarrow \exists d \in DS$ tel que $(d,s) \in$ state_mapping
Si un état physique (d) peut être affecté par une action système (a), alors il est possible qu'un état logique correspondant à cet état physique (d) soit à l'origine de cette action :
$\forall a \in SA, d \in DS, d \in \text{dom}(\text{sys-action}(s)) \Rightarrow \exists s \in LS$ tel que $(d,s) \in$ state_mapping
Si l'une de ces deux propositions ne sont pas vérifiées, alors cela signifie que les possibilités physiques du dispositif ne sont pas toutes exploitées pour l'interaction, ou que qu'un état du système est ignoré. Dans ces deux cas, les concepteurs doivent vérifier que ce n'est pas une erreur de conception.

En résumé, le modèle de la physicalité introduit une approche formelle basée sur des automates pour décrire la relation l'état physique et l'état logique d'une interface. De nouvelles caractéristiques, comme le rebond, sont identifiées, tandis que des changements dans l'automate (transitions) sont initiés par l'utilisateur ou par le système. En proposant une approche formelle liant état physique et état logique, ces travaux traite les enjeux de gestion des conflits entre les mondes physique et numérique (Chapitre 2, section 3.2.2).

2. Analyse des outils existants au regard des objectifs

Nous rappelons que l'objectif de nos travaux est l'aide à la conception des interfaces mixtes. Notre participation au domaine vise un espace de conception afin de comprendre, de cerner, et de contribuer à l'espace des possibilités des systèmes de réalité mixte. Pour cela, nous avons détaillé nos objectifs :

1. Capitaliser et englober les résultats validés par la communauté scientifique concernant la conception des systèmes de réalité mixte,
2. Définir une approche de conception qui exploite de façon systématique les concepts identifiés dans notre espace.

Un outil conceptuel qui remplit ces objectifs doit prendre en compte les questions de recherche liées aux interfaces mixtes qui ont été introduites au chapitre précédent. Au fil de ce chapitre, nous avons souligné comment les travaux conceptuels existants étaient reliés à ces enjeux.
Nous analysons maintenant les outils conceptuels existants en dégageant leurs dimensions utiles à la conception, puis leurs chevauchements, pour enfin mettre en évidence leur incomplétude.

2.1. Dimensions utiles

Comme premier pas vers la définition d'un espace de conception intégrateur, il convient d'identifier dans les travaux exposés les dimensions utiles à la conception, ainsi que leurs chevauchements et leurs manques.
Pour identifier les dimensions utiles à la conception, nous cherchons la pertinence des caractéristiques de ces travaux en vérifiant qu'elles engendrent un espace de conception couvrant au moins les exemples de systèmes de réalité mixte existants. Pour éviter le biais de la recherche, ces exemples sont trouvés dans la littérature scientifique du domaine, mais aussi conçus par d'autres domaines (designers, artistes, etc.). Nous reprenons pour cela les quatorze exemples présentés au Chapitre 2, qui couvrent un large éventail de l'espace de conception. Nous avons pu illustrer les caractéristiques existantes présentées dans ce chapitre sur ces systèmes. Ceci montre que les caractéristiques issues de ces travaux ne sont pas artificielles et nécessitent d'être capitalisées.

2.2. Chevauchements

Nous venons de voir que les travaux existants sont pertinents pour classer les systèmes que nous avons pris comme exemples. Cependant, l'espace de conception engendré par la fusion des approches existantes n'est pas parfait, puisqu'il existe des chevauchements entre les différents travaux présentés. Parmi les recouvrements possibles entre approches, nous identifions les suivants :

- De nombreux travaux caractérisent les objets mis en jeu dans l'interaction : ils peuvent être objet de la tâche ou outil : ASUR [Dubois, 2001] avec les composants S_{tool} et R_{tool} d'un côté pour les outils, et S_{object} et R_{object} de l'autre côté pour les objets de la tâche ; IRVO [Chalon, 2004] avec les éléments O et T ; [Underkoffler, Ishii, 1999] et les objets verbes ; et encore [Holmquist et al., 1999] avec les outils qui s'opposent aux objets.
- Le mélange des deux mondes physique et numérique a été caractérisé dans plusieurs travaux. Le continuum de [Milgram, Kishino, 1994] est réutilisé sous un autre nom (représentation de l'interaction) dans [Fitzmaurice et al., 1995]. Il est redéfini plus précisément par [Dubois, 2001]. Cette nouvelle caractérisation est capitalisée dans IRVO [Chalon, 2004] pour caractériser les objets de la tâche et les outils (avec les indices r, r+v, v+r, v).
- Nous trouvons une caractérisation implicite ou explicite de la modalité d'interaction, à plusieurs niveaux : ASUR [Dubois, 2001] et ses composants adaptateurs d'entrée/sortie, et les caractéristiques des relations (nombre de dimensions, type de langage [Dubois, 2001], mécanisme de capture et représentation [Dubois, Gray, 2007]). De même, l'axe « entrée et sortie » de [Fitzmaurice et al., 1995] caractérise entre autre la modalité. Dans d'autres travaux, nous trouvons encore ce type de caractéristiques sous le terme de « medium » et « adaptateurs » dans [Benford, Fahlen, 1993], sous le terme de « mouvements capturés » et leurs caractéristiques (degré de liberté, précision, etc.) dans [Benford et al., 2005]. Tous ces éléments font écho aux recherches sur l'interaction multimodale.
- La dynamicité du lien entre physique et numérique est caractérisée de la même façon dans plusieurs travaux. Dans [Underkoffler, Ishii, 1999] et [Holmquist et al., 1999], nous trouvons une distinction entre objets liés de façon statique ou dynamique (respectivement appelés outils reconfigurables et conteneur). Le couplage entre pyfo et variable est caractérisé dans [Shaer, 2004] par deux possibilités : le couplage statique est défini comme celui qui intervient en conception ; le couplage dynamique est défini comme celui qui intervient pendant l'utilisation. Dans [Fitzmaurice et al., 1995], nous trouvons que l'assignation d'une fonction informatique à une brique pouvait être permanente, programmable, ou re-assignée rapidement. Dans [Renevier, 2004], cette dynamicité est précisée sous la forme de deux caractéristiques indépendantes : (1) mode d'interaction (passif ou actif) lors de la création du lien, et (2) la temporalité du lien (éphémère ou persistant).
- La composition spatiale entre les éléments physiques et numériques est caractérisée dans [Fitzmaurice et al., 1995] comme superposés ou non, dans [Benford, Fahlen, 1993] via le

focus et la nimbus, et dans [Fishkin, 2004] par l'incarnation. Enfin c'est cette même composition spatiale entre les éléments physiques et numériques qui est caractérisée par la continuité perceptuelle de [Dubois, 2001].

Que ces chevauchements soient dûs à l'extension intentionnelle d'une caractéristique d'une proposition à l'autre, ou bien à la capitalisation implicite ou explicite d'une dimension, ces chevauchements ne favorisent pas la lisibilité du domaine. En effet, une même caractéristique a parfois des noms différents d'une proposition à l'autre. De plus cette forme de capitalisation est partielle et n'aborde qu'une partie du problème global de conception.

2.3. Incomplétude

Les exemples que nous avons choisis, pour montrer la pertinence des travaux existants et la nécessité de les capitaliser, couvraient un éventail très large d'interfaces mixtes. Leur diversité facilite donc la classification. Pour montrer qu'il existe des manques dans la réunion de ces espaces de conception, nous choisissons au contraire des systèmes similaires, plus difficiles à distinguer. Pour cela, nous reprenons le scénario du DigitalDesk avec le bouton *FILL* de papier et la gomme (section 2.1.1.1 du Chapitre 2), le système PICO avec les palets (section 2.1.1.3 du Chapitre 2). Nous enrichissons l'ensemble des exemples de cinq interfaces mixtes :

- NavRNA [Bailly, 2006] est un système pour interagir avec les molécules d'ARN. Comme le montre la Figure 45 (a), les biologistes sont réunis autour d'une table équipée d'une caméra et d'un projecteur. La caméra capture la position des jetons bleus que l'utilisateur manipule pour explorer (c'est-à-dire déplacer, tourner, redimensionner) la vue en deux dimensions de la molécule d'ARN.
- Le Phicon [Ullmer, Ishii, 1997] signifie icône physique. Dans le *Tangible Geospace* [Ullmer, Ishii, 1997] le Phicon est un outil qui a la forme du grand dôme du MIT (Figure 45 (b)). Les utilisateurs manipulent le Phicon sur la table, sur laquelle une carte du campus du MIT est projetée. La position de Phicon sur la table est à chaque instant la même que la position du dôme sur la carte.
- La reacTable [Jordà et al., 2007] est utilisée comme un synthétiseur musical, où les cubes et les jetons représentent les modules du synthétiseur (Figure 45 (c)). Les utilisateurs peuvent toucher directement la surface avec leurs doigts pour interagir. Ils peuvent également changer les distances et orientations relatives entre les objets pour contrôler le synthétiseur. Pour ce système, nous considérons dans notre étude l'interaction la plus proche des autres exemples, c'est-à-dire l'interaction avec les objets. La table est équipée d'une caméra qui capte la nature, la position et l'orientation des objets, et d'un projecteur pour afficher une animation sur la surface correspondant à l'état interne des objets.



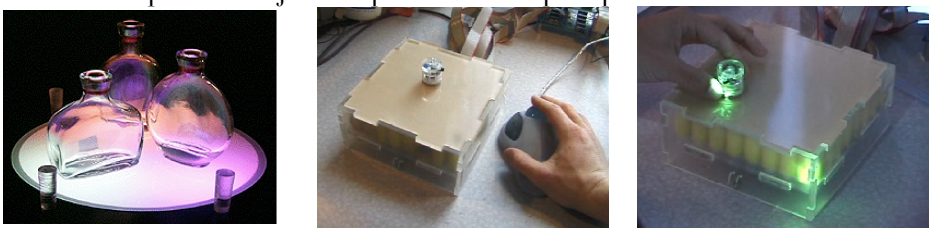
(**a**) (**b**) (**c**)

Figure 45: NavRNA (a), le Phicon du grand dôme du MIT dans le *Tangible Geospace* (b), et la reacTable (c).

- Les *music bottles*[23] [Ishii et al., 2001] sont des objets qui permettent d'interagir avec un lecteur de musique. Chaque bouteille contient une voix du morceau musical. Quand l'utilisateur pose la bouteille sur la table et l'ouvre (Figure 46 (a)), la voix correspondante est jouée. De plus, de la lumière colorée correspondant à la hauteur et au volume du son est projetée par-dessous la table (Figure 46 (a)).

[23] Bouteilles musicales

- L'*actuated workbench*[24] [Pangaro et al., 2002] est une table équipée d'aimants contrôlables par le système. Sur cette table, l'utilisateur, ou le système, peuvent manipuler les objets. La manipulation d'un objet peut être indirecte, en utilisant une trackball (Figure 46 (b)), ou directe en manipulant l'objet si la position est captée par caméra.



(**a**) (**b**) (**c**)

Figure 46: *Music bottles* (a), l'*Actuated workbench* utilisé avec une souris (b) ou augmenté grâce à la vision par ordinateur (c).

Ces exemples ont été choisis à dessein, car ils paraissent très similaires : tous permettent la manipulation d'objets tangibles sur une surface plane. Néanmoins ces exemples définissent des interactions différentes. Il convient de capturer des différences par des dimensions de conception. Nous analysons ces exemples suivants les approches existantes pour montrer qu'elles ne permettent pas de capturer ces différences. Nous présentons maintenant cinq manques identifiés dans les travaux présentés.

Nous constatons que plusieurs interfaces mixtes exploitent plusieurs modalités, comme les *music bottles* avec la projection sur la table et la diffusion du son. La multiplicité des modalités (en entrée ou en sortie) n'est pas capturée par les approches existantes. Nous avons pourtant vu que répondre à la multiplication des possibilités était un des enjeux des systèmes de réalité mixte.
Un autre enjeu réside dans la recherche d'interfaces plus intuitives, et les concepteurs ont proposé plusieurs façons de répondre à ce problème, par exemple par l'utilisation de métaphores. Or en l'état, comment distinguer finement les différents types de métaphores ? Par exemple, les métaphores du bouton de papier *FILL* et de la gomme du DigitalDesk font appel à deux cultures différentes, même si toutes les deux sont des métaphores de verbe [Fishkin, 2004]. De même, la métaphore de nom de la gomme du DigitalDesk n'est pas exactement du même type que la métaphore de nom du Phicon en forme de dôme du MIT du Tangible Geospace. Dans un cas, la métaphore fait référence à la commande (gommer) et dans l'autre au paramètre de cette commande (déplacer le dôme). Les approches présentées dans ce chapitre ne permettent pas de faire cette différence.
Les travaux existants donnent une place importante aux contraintes en considérant ce qu'elles induisent pour l'entrée du système. Or, il serait utile pour le concepteur de distinguer aussi les contraintes au regard de la sortie : par exemple, dans la reacTable ou les *music bottles*, l'affichage des animations ne peut pas être occulté par la main des utilisateurs. Au contraire dans PICO ou l'*actuated workbench*, les utilisateurs peuvent occulter la projection avec leur main. Considérer les contraintes au regard de la sortie permet aux concepteurs d'anticiper certains problèmes d'interaction.
Nous constatons aussi l'absence de caractérisation de la capacité de l'utilisateur à modifier une propriété de son outil. Par exemple avec la reacTable (Figure 47), l'utilisateur peut ajuster les propriétés de ses outils grâce à un autre type d'outil : dans la Figure 47, un oscillateur à basse fréquence contrôle un filtre sonore passe-bas. Au contraire les objets manipulés dans NavRNA ou le Tangible Geospace ne peuvent pas voir leurs propriétés ajustées. Cette caractéristique est complémentaire à celle de dynamicité du lien qui a beaucoup été caractérisée [Underkoffler, Ishii, 1999] [Holmquist et al., 1999] [Fitzmaurice et al., 1995] [Shaer, 2004] [Renevier, 2004], et qui signifie que l'utilisateur ou le système peut changer complètement la fonction numérique de l'outil. Ici il s'agit seulement d'ajuster des paramètres de l'outil, comme on réglerait l'ouverture d'une clé anglaise grâce à sa crémaillère : sa fonction ne change pas.

[24] Plan de travail avec effecteur

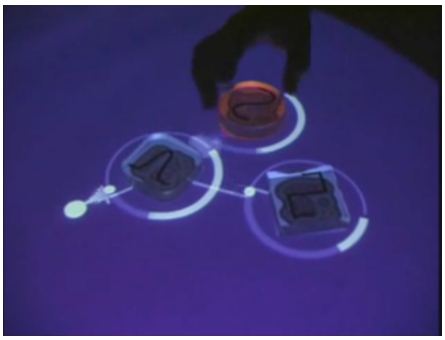


Figure 47: Un outil de la reacTable dont une propriété est contrôlée par un autre outil : Le premier est un filtre sonore passe-bas (objet bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus). Le second est un oscillateur à basse fréquence (objet circulaire et rouge avec une sinusoïde représentée dessus).

Enfin, il manque dans les travaux existants la différence du point de vue de l'utilisateur entre les différents niveaux de retour d'information, autrement que par la modalité utilisée comme avec les boucles B et C de [Dix et al., 2007] à la Figure 41. Ceci fait partie de l'enjeu cité au chapitre précédent à propos de la place laissée au contrôle par l'utilisateur : comment distinguer le retour d'information direct de l'objet de la tâche vers l'utilisateur, comme le retour d'information direct de l'outil vers l'utilisateur, par rapport au retour d'information indirect de l'objet de la tâche vers l'outil, qui permet à l'utilisateur de réajuster son action sur l'outil ?

3. Synthèse

Dans ce chapitre, nous avons étudié les approches et espaces de conception existants pour les interfaces mixtes. De cette étude de l'état de l'art, nous retenons un ensemble de dimensions utiles à la conception. Nous notons aussi que certaines dimensions ont été étudiées très en détail et se retrouvent décrits par des termes différents selon les approches. Enfin, en considérant des exemples existants, nous avons montré que ces approches ne couvrent pas tous les aspects de conception puisque des systèmes existants aux interactions différentes ne sont pas distingués. Ceci montre que les travaux existants n'ont pas encore dévoilé la totalité des dimensions de conception.
Enfin, cet état de l'art souligne aussi la diversité des approches existantes et leur complémentarité. Néanmoins, utiliser l'ensemble de ces approches pour concevoir un système de réalité mixte nécessite de jongler difficilement entre tous ces éléments non homogènes.
Face à ce constat, nous présentons le modèle d'interaction mixte au chapitre suivant, qui constitue une approche intégratrice de ces travaux. La pertinence de notre modèle, outre son aspect intégrateur, doit passer par sa capacité à dégager de nouvelles dimensions de conception pour combler les manques des approches existantes.

Chapitre 4 : La solution proposée, le modèle d'interaction mixte

Dans ce chapitre, nous introduisons un nouveau modèle d'interaction, le *modèle d'interaction mixte*. Un modèle d'interaction, tel qu'il est défini dans [Beaudouin-Lafon, 2004] ou [Appert, 2007], est une description du déroulement de l'interaction. Il est utilisé en conception pour explorer l'espace des possibilités. En effet, le but d'un modèle d'interaction est de fournir un cadre de travail aux concepteurs d'un système interactif.
Un modèle d'interaction diffère des règles ergonomiques, des modèles au sens de l'ingénierie dirigée par les modèles et des modèles d'architecture logicielle :

- Les règles ergonomiques aident la conception de l'interaction en évaluant un système déjà conçu pour le corriger, alors qu'un modèle d'interaction est utilisé très tôt, pendant les premières étapes de conception.
- Un modèle, au sens de l'ingénierie dirigée par les modèles, est une expression de l'information structurée qui représente des aspects du système. Les modèles, comme UML, sont largement utilisés en génie logiciel pour le développement. Ils sont utilisés dans l'industrie pour analyser le problème, pour communiquer entre participants, pour documenter le système ou pour générer du code. Au contraire, un modèle d'interaction est une description de l'interaction indépendante des aspects de réalisation logicielle destinée à être utilisée très tôt dans la conception, pendant la phase exploratoire et expérimentale.
- Pour envisager l'utilisation du modèle d'interaction mixte dans un processus de développement dirigé par les modèles, il faut avoir défini un langage de modélisation qui permet d'interpréter la signification des composants de la structure. La définition d'un tel langage constitue une perspective de nos travaux. Avant cela, nous identifions dans ce chapitre les concepts pertinents pour la conception.
- Un modèle d'architecture logicielle est une architecture logicielle de référence. Parmi les modèles d'architecture logicielle pour l'Interaction Homme-Machine, nous trouvons MVC, PAC ou Arch. Une architecture logicielle est en ensemble organisé de composants, de relations et de principes directeurs (norme IEEE 1471). Elle définie l'organisation fondamentale du logiciel. Elle s'adresse aux développeurs et permet la communication, la retro-conception d'un système existant et l'évaluation du logiciel selon des critères de qualité logicielle. Au contraire, un modèle d'interaction est une description de l'interaction indépendante des aspects de réalisation logicielle destinée aux concepteurs avec leurs expertises diverses (Design, Informatique, Ergonomie, etc.).

Dans [Beaudouin-Lafon, 2004], trois dimensions sont définies pour évaluer un modèle d'interaction :

- Son pouvoir descriptif, c'est-à-dire sa capacité à décrire un large éventail d'interfaces,
- Son pouvoir évaluatif/comparatif/taxinomique, c'est-à-dire sa capacité à aider l'évaluation, la comparaison et le classement des interfaces,
- Son pouvoir génératif, c'est-à-dire sa capacité à aider les concepteurs à générer des techniques d'interaction. Il s'agit ici d'être capable d'explorer l'espace de conception.

Dans une première partie, nous décrirons l'entité centrale à notre modèle, un objet mixte, puis l'interaction mettant en jeu de tels objets mixtes. Nous démontrerons le pouvoir descriptif du modèle d'interaction mixte grâce aux exemples introduits au Chapitre 2.
Nous étudions ensuite dans la seconde partie son pouvoir comparatif grâce aux exemples plus difficiles à distinguer introduits dans le Chapitre 3. Pour cela, nous considérons d'abord l'objet mixte indépendamment du contexte applicatif (caractéristiques intrinsèques). Nous considérons ensuite l'objet mixte dans son contexte applicatif (caractéristiques extrinsèques).

1. Modèle d'interaction mixte : centrer l'interaction entre l'utilisateur et le système autour des objets mis en jeu

1.1. Motivations

Pour décrire l'interaction entre l'utilisateur et le système, nous prenons, comme éléments centraux entre l'utilisateur et le système, les objets mis en jeu dans l'interaction. Nous basons notre définition sur la définition courante d'un objet :

> **Objet** n.m. **1.** Ce qui affecte les sens, spécialement la vue. **2.** Chose généralement maniable, destinée à un usage particulier. […] **5.** Ce qui occupe l'esprit, ce à quoi s'applique l'esprit, l'entendement ➨ PHILO La chose même qui est pensée, par opposition au sujet qui pense. […]

Définition 1 : Objet (source : Dictionnaire Hachette).

Nous trouvons dans la littérature des travaux qui vont également dans le sens d'une approche « objet d'interaction » , comme [Beaudouin-Lafon, 2004] [Dourish, 2001] [Fishwick, 2008]. D'autres travaux (Chapitre 3) ne mettent pas l'accent sur les objets mis en jeu dans l'interaction, mais focalisent sur le flux d'information entre les objets. Ces deux approches sont évidemment très complémentaires.
En prenant les objets comme centre de la description de l'interaction, nous visons à fournir plusieurs niveaux d'analyse : niveau objet et niveau interaction. De plus, nous verrons que notre approche permet de baser la conception sur des caractéristiques intrinsèques et extrinsèques de l'objet mixte. Pour les concepteurs, ces deux ensembles de caractéristiques rendent possible la réutilisation de leurs choix pour des contextes applicatifs différents : en effet les caractéristiques intrinsèques restent inchangées d'un contexte applicatif à l'autre, alors que les caractéristiques extrinsèques changent.

Nous présentons tout d'abord notre définition d'un objet d'interaction dans une interface mixte, appelé objet mixte ou objet hybride (niveau objet), puis nous présentons notre modèle de l'interaction avec de tels objets (niveau interaction).

1.2. Description d'un objet mixte

1.2.1. Fondements

Nous considérons un objet physique comme un ensemble de propriétés physiques, comme sa texture, sa température, ses couleurs, son poids, etc. Ces propriétés physiques peuvent affecter les sens (Définition 1 (**1.**)). Il peut aussi être maniable (Définition 1 (**2.**)), mais pas nécessairement (Définition 1 (**5.**)).
Nous considérons un objet numérique comme un ensemble de propriétés numériques, appartenant donc au monde de l'informatique. Ces propriétés sont des abstractions, et font écho à la définition **5.** de l'objet. Elles peuvent être par exemple la représentation numérique d'une image, une propriété booléenne (vraie ou fausse), etc. La Figure 48 schématise cette idée de base pour les objets physiques et numériques sous forme d'ensemble de propriétés.

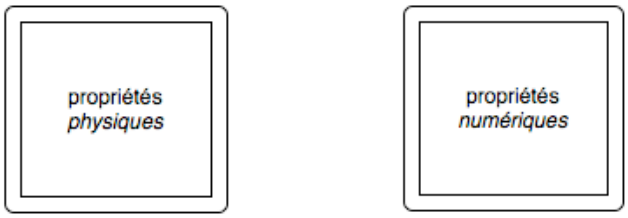


Figure 48 : Description générique des objets physiques (a) et numériques (b).

Pour définir un objet mixte, nous nous appuyons sur ces définitions. Nous définissons un objet mixte ou hybride comme deux ensembles de propriétés, physique et numérique, qui sont reliés. Cette liaison fait partie intégrante de l'objet mixte. La Figure 49 représente cette définition préliminaire d'un objet mixte.

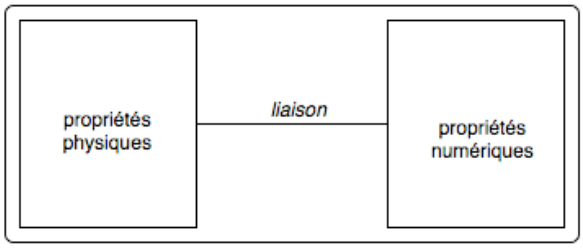


Figure 49 : Description générique d'un objet mixte.

L'originalité de notre modèle réside dans la définition de la liaison entre les ensembles de propriétés. Pour définir la liaison, nous utilisons la définition d'une modalité d'interaction issue de travaux sur de l'interaction multimodale [Nigay, 1994]. Une modalité d'interaction *m* y a été définie formellement, en notant :

d le dispositif physique d'interaction, en entrée ou en sortie, c'est-à-dire l'objet qui permet l'acquisition ou le rendu de l'information par le système,

l le langage d'interaction, un ensemble d'expressions bien formées, composées de symboles, et qui ont un sens.

Une modalité *m* est définie alors de la façon suivante :

$$m \rightarrow (d, l) \mid (m, l)$$

Par exemple, *(clavier, langage pseudo-naturel)* est une modalité d'interaction.

1.2.2. Modèle d'un objet mixte

1.2.2.1. Description

Grâce au formalisme de description d'une modalité, nous explicitons (Figure 50) le lien entre propriétés physiques et numériques selon deux sens, entrée et sortie, ainsi qu'avec deux niveaux d'abstraction, dispositif et langage, grâce à la notion de **modalité de liaison** :

En entrée, une modalité de liaison peut :

a. Capter certaines propriétés physiques grâce à un **dispositif de liaison en entrée**,
b. Interpréter ces données physiques captées grâce à un **langage de liaison en entrée**, pour constituer une partie des propriétés numériques de l'objet mixte.

En sortie, une modalité de liaison peut :

c. Générer des données à partir des propriétés numériques de l'objet mixte, grâce à un **langage de liaison en sortie**,
d. Traduire ces données générées en propriétés physiques observables par l'utilisateur grâce à un **dispositif de liaison en sortie**.

Une modalité de liaison est donc différente d'une modalité d'interaction par son niveau d'abstraction. D'un côté, la modalité de liaison définit la liaison entre les deux ensembles de

propriétés d'un objet mixte. De l'autre côté, une modalité d'interaction est définie comme une modalité dont le dispositif est un objet mixte. Nous expliquerons plus en détail dans la section 1.3 comment nous intégrons la définition de modalité d'interaction dans la description de l'interaction. La modalité de liaison est intrinsèque à l'objet et ne dépend pas de l'utilisation qui sera faite de cet objet.

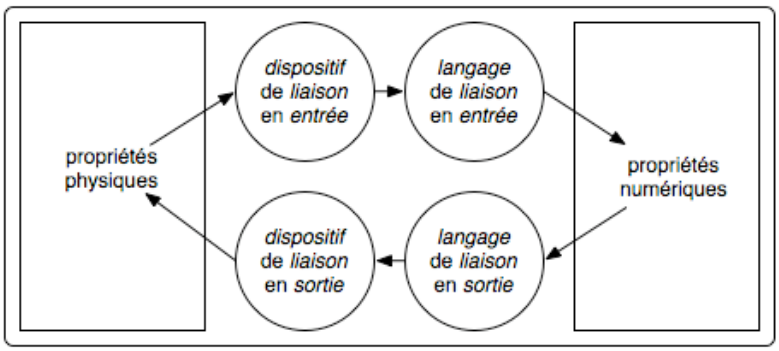


Figure 50 : Description générique d'un objet mixte simple avec une modalité de liaison en entrée et une modalité de liaison en sortie, chacune étant composée d'un dispositif et d'un langage.

1.2.2.2. Justification

Nous aurions pu définir cette liaison à l'aide de deux niveaux d'abstraction (physique/numérique) ou de plus de niveaux d'abstraction. Le cas de deux niveaux d'abstraction se trouve déjà dans la littérature sur les systèmes de réalité mixte (par exemple dans [Dix et al., 2007]). Le lien entre ces deux niveaux d'abstraction n'est souvent pas mis en évidence et il est utile de l'expliciter pour la conception. D'autres approches ont donc cherché à l'expliciter, par exemple avec un niveau d'abstraction de plus comme dans [Dubois, 2001]. Mais ne détailler le lien qu'à travers le niveau Adaptateur [Dubois, 2001] ne révèle pas les différentes possibilités de conception de l'interaction avec cet adaptateur. Par exemple pour les interfaces graphiques, l'écran peut être utilisé pour afficher de l'information de plusieurs façons différentes (texte, graphique). Nous avons choisi de décrire cette forme entre le niveau dispositif et le niveau numérique par un niveau d'abstraction, noté langage de liaison.
Plus de quatre niveaux d'abstraction expliciteraient plus précisément la façon dont le système interpréterait l'information. Pourtant, nous avons choisi de décrire cette transformation dans un seul niveau d'abstraction correspondant au langage de liaison. En effet, l'espace de conception lié au niveau d'abstraction du langage est suffisamment riche (section 2.1.1.2 de ce chapitre). Détailler le niveau langage sera utile au moment du développement logiciel afin d'identifier des blocs élémentaires de transformation des données (Chapitre 7).

1.2.2.3. Niveau d'abstraction

Cette définition de la liaison en deux niveaux d'abstraction nous permet également de composer les objets mixtes. En effet, une modalité *m* peut être élémentaire, sous la forme *(d, l)* ou elle peut être composée, sous la forme *(m, l)*, comme par exemple *((caméra, seuil), détection de la main)*. Ainsi, de la même façon, un objet mixte peut être décrit comme un objet mixte ou comme le dispositif d'une modalité de liaison.
Prenons l'exemple de l'écran : il peut être dispositif de liaison en sortie (Figure 51, haut), mais aussi un objet mixte à part entière (Figure 51, bas). Dans la plupart des cas, les concepteurs n'auront pas besoin de concevoir l'écran et donc de le modéliser en tant qu'objet mixte. C'est l'exemple de CASPER (Figure 51, haut) qui utilise un écran comme dispositif de liaison en sortie pour l'aiguille de ponction. Néanmoins pour certains cas comme l'exemple de la Figure 51 (bas), des objets conçus « sur mesure » peuvent nécessiter de détailler l'écran comme objet mixte. C'est l'exemple de *White Snow* [Campbell, 2001], une installation artistique qui présente en lieu public une grande image d'un personnage courant dans la neige (Figure 51, bas). Pour cette installation, l'artiste a construit l'écran sur mesure avec une matrice de 16 × 12 (192) ampoules de 50 Watts.

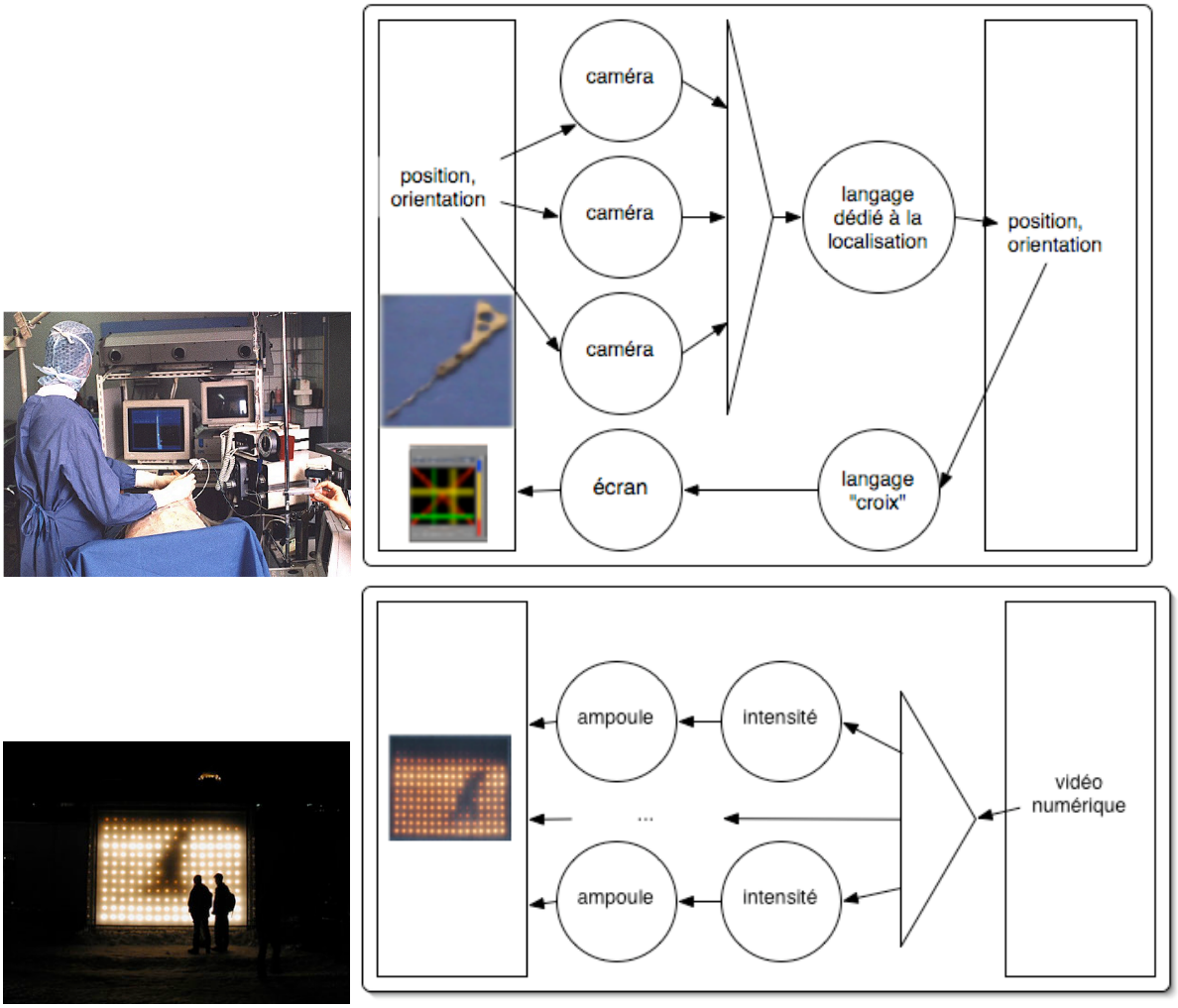


Figure 51 : L'écran modélisé comme un dispositif d'objet mixte (en haut) ou comme objet mixte (en bas). En haut, CASPER est un système de réalité augmentée pour la chirurgie. En bas, *White Snow* est une installation artistique de Jim Campbell [Campbell, 2001].

En synthèse, la propriété de composition de notre modèle d'interaction mixte autorise le concepteur à choisir le niveau d'abstraction qui lui convient.

1.2.2.4. Objets mixtes multimodaux

La réutilisation des travaux en interaction multimodale pour définir la modalité de liaison d'un objet nous permet aussi de décrire précisément les objets avec des liaisons multimodales. Nous réutilisons les travaux sur la composition des modalités d'interaction [Vernier, Nigay, 2000] [Nigay, Coutaz, 1995] pour décrire la composition des modalités de liaison. La Figure 52 présente l'exemple d'un objet mixte avec une liaison en entrée multimodale : l'aiguille de ponction de CASPER [Dubois, 2001]. Dans cet exemple, l'acquisition de la position et l'orientation de l'aiguille du chirurgien repose sur la fusion du flux vidéo de trois caméras. La fusion est ici faite au niveau des dispositifs de liaison en entrée. La Figure 53 présente l'exemple d'un objet avec liaison multimodale en sortie : une *music bottle*[25] [Ishii et al., 2001]. Dans cet exemple, la matérialisation de la partie musicale à l'intérieur de la bouteille repose sur la fission de l'information en sortie. La fission est ici faite au niveau des langages de liaison en sortie.

[25] bouteille musicale

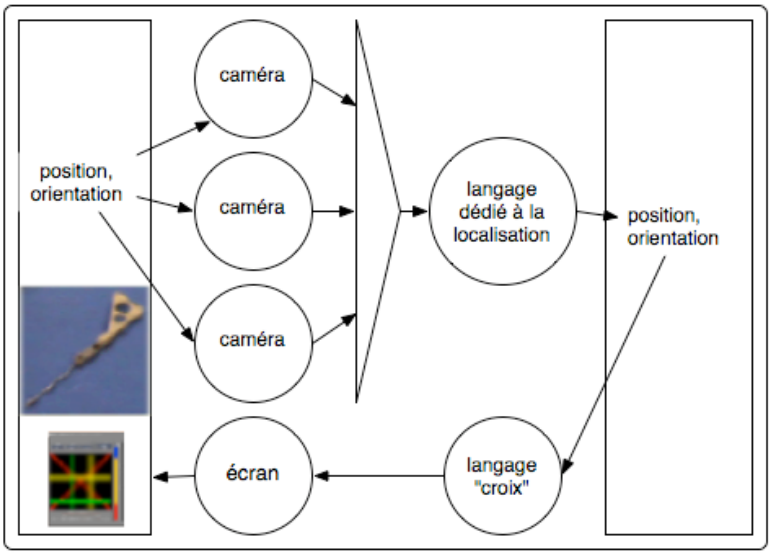


Figure 52 : Exemple d'un objet mixte avec une liaison multimodale en entrée : l'aiguille de ponction du chirurgien dans CASPER [Dubois, 2001].

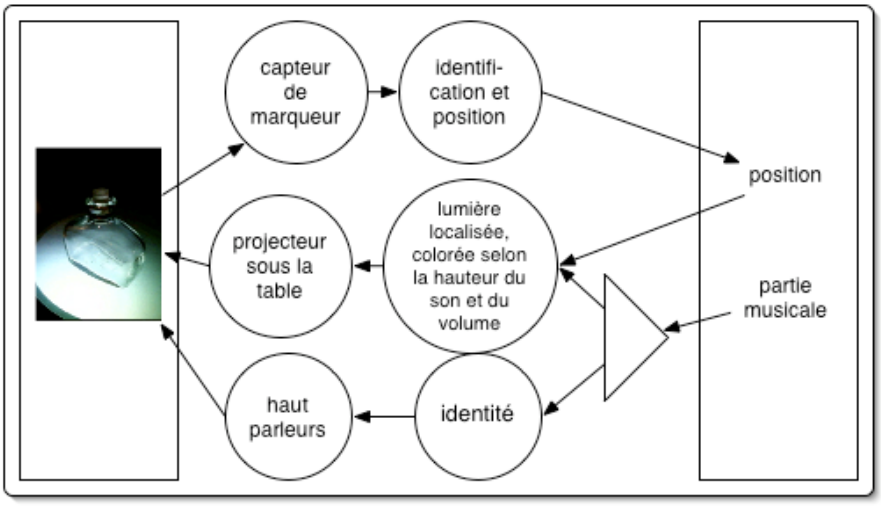


Figure 53 : Exemple d'un objet mixte avec une liaison multimodale en sortie : une *music bottle* [Ishii et al., 2001].

Nous venons de présenter la modalisation d'un objet mixte selon deux ensembles de propriétés physiques et numériques et des modalités de liaison. Nous décrivons maintenant l'interaction avec un objet mixte.

1.3. Description de l'interaction avec des objets mixtes

1.3.1. Fondements

Nous basons notre description de l'interaction entre l'utilisateur et le système interactif mettant en jeu des objets mixtes sur le modèle d'interaction instrumental [Beaudouin-Lafon, 2004]. La Figure 54 présente le modèle d'interaction instrumental appliqué à une interaction graphique : l'utilisateur navigue dans le document via la souris et une barre de défilement.

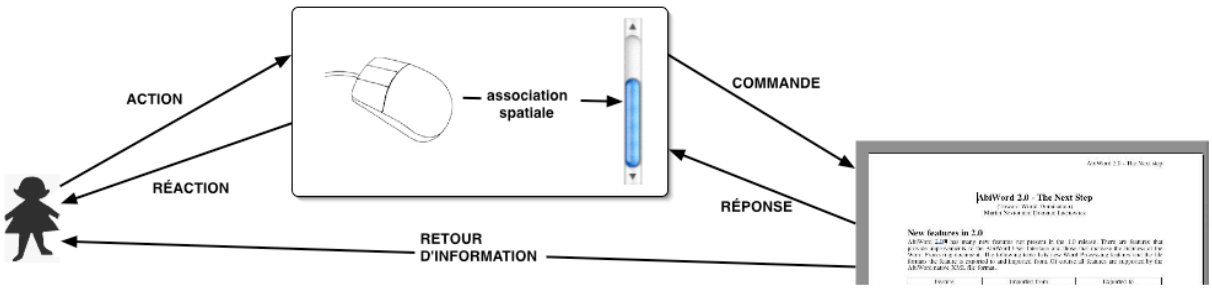


Figure 54 : Le modèle d'interaction instrumental appliqué à une interaction graphique : l'utilisateur navigue dans le document via la souris et une barre de défilement.

Dans ce modèle d'interaction illustré sur les interfaces graphiques, l'utilisateur (à gauche dans la Figure 54) agit sur l'instrument (cadre central de la Figure 54, appelé outil dans d'autres travaux [Dubois, 2001][Holmquist, 1999][Underkoffler, Ishii, 1999]). L'instrument transforme l'action en commande ciblée sur l'objet d'intérêt du domaine (à droite dans la Figure 54, appelé objet de la tâche dans [Dubois, 2001]).

La réaction de l'instrument permet à l'utilisateur de contrôler son action sur l'instrument (curseur de la souris). L'objet du domaine fournit une réponse vers l'instrument qui la transforme pour qu'elle participe à la réaction (position de la barre de défilement). La réaction de l'outil est composée de la mise à jour de la position du pointeur de la souris et de la mise à jour de la position de la barre de défilement dans son rail. Cette deuxième réaction est rendue possible grâce à la réponse de l'objet de la tâche vers l'outil.
Il y a de plus un retour d'information direct de l'objet de la tâche vers l'utilisateur. En effet, dans l'exemple de la barre de défilement, l'utilisateur voit directement la position dans le document à travers la fenêtre, en plus de la voir grâce à la barre de défilement : Il y a donc deux retours : la réponse et le retour d'information.
Les trois types de réaction participent à l'observabilité du système et permettent à l'utilisateur d'évaluer si ses actions ont bien eu l'effet escompté.

L'instrument lui-même se compose d'une partie logique (par exemple, la barre de défilement, Figure 54) et d'une partie physique (par exemple la souris, Figure 54). Ce point de vue sur l'instrument est à rapprocher de la définition d'une modalité comme un couple (d, l) et de la définition de la réalité mixte (dispositif physique et traitement informatique), mais aussi du multiplexage temporel de l'interaction : une seule souris permet d'utiliser plusieurs fonctions, dont la barre de défilement. L'activation d'un instrument se produit quand les parties logique et physique ont été associées.
L'association entre la partie physique et logique de l'instrument peut être :

- **Spatiale** : L'utilisateur apporte la partie physique dans l'espace d'une partie logique. C'est le cas de la Figure 54 car l'utilisateur qui apporte le curseur de la souris sur la barre de défilement.
- **Temporelle** : L'utilisateur a activé l'instrument grâce à une action précédente. C'est le cas lorsque l'utilisateur sélectionne l'outil « main » ( ) dans les logiciels de type Acrobat Reader pour ensuite faire défiler le document.

1.3.2. Modèle d'interaction mixte

1.3.2.1. Description

Pour construire le modèle d'interaction mixte, nous avons enrichi le modèle d'interaction instrumental par l'ajout de la notion de modalité d'interaction [Nigay, 1994]. Une modalité est alors un instrument (dispositif) et un langage d'interaction.

Nous considérons ensuite les objets mixtes dans ce modèle d'interaction instrumental enrichi. Ceux-ci sont soit des instruments, notés outils mixtes, soit l'objet de la tâche [Dubois, 2001]. Ainsi l'outil mixte est dispositif d'une modalité d'interaction, et nous lui associons un langage d'interaction qui traduit l'action de l'utilisateur en tâche élémentaire (commande). L'objet de la tâche mixte est manipulé par l'utilisateur via la modalité d'interaction. L'ajout du langage d'interaction au modèle instrumental permet de préciser la description de l'interaction entre outil et objet de la tâche : en effet, nous verrons qu'un outil peut être utilisé de différentes façons pour agir sur l'objet de la tâche. Le langage d'interaction permet de décrire cette interaction.
La Figure 55 illustre le modèle d'interaction mixte sur un cas générique simple qui décrit l'interaction entre l'utilisateur et le système via un outil mixte et un objet mixte de la tâche. Le modèle d'interaction mixte permet de décrire l'interaction à plusieurs niveaux d'abstraction. À la Figure 55, à haut niveau d'abstraction, l'utilisateur fait une action sur l'outil mixte, qui réagit. Cette action sur l'outil mixte est transformée par le langage d'interaction en entrée en tâche élémentaire qui s'applique à l'objet de la tâche. Nous supposons ici que l'objet de la tâche est mixte lui aussi dans le cadre de l'interaction avec les systèmes de réalité mixte. L'objet de la tâche répond à l'outil, via le langage d'interaction en sortie. Cette réponse est prise en compte pour ajuster la réaction de l'outil. L'objet de la tâche adresse aussi un retour d'information à l'utilisateur.

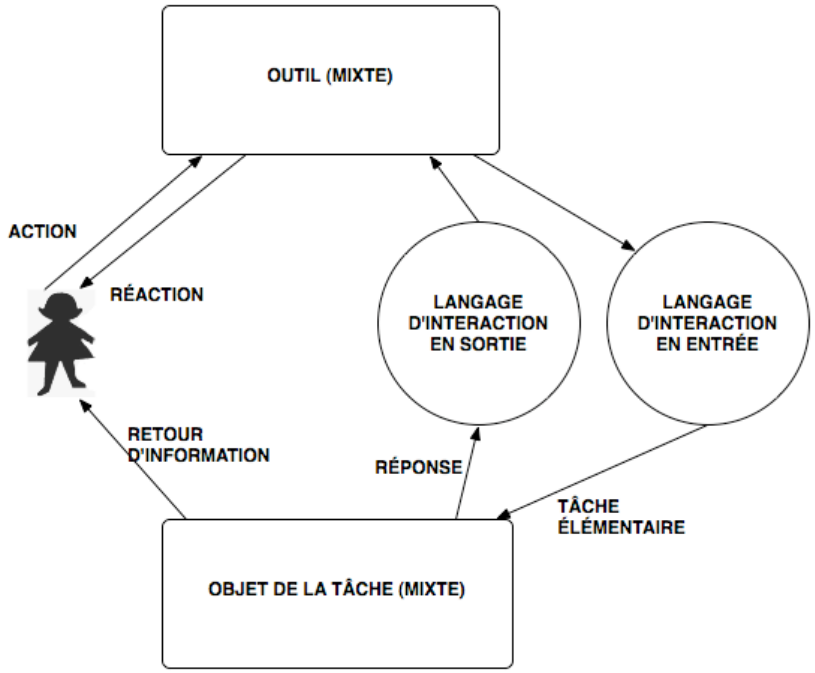


Figure 55 : Description générique d'une interaction simple à un haut niveau d'abstraction grâce au modèle d'interaction mixte.

À la Figure 56, l'interaction est décrite à un plus bas niveau d'abstraction en détaillant les objets mixtes mis en jeu dans l'interaction : l'utilisateur fait une action sur les propriétés physiques de l'outil mixte. Cette modification des propriétés physiques de l'outil mixte entraîne la mise à jour de ses propriétés numériques, via sa modalité de liaison en entrée. Ce nouvel état interne de l'outil est :

1. rendu observable grâce à la modalité de liaison en sortie : des propriétés physiques de l'outil sont mises à jour elles aussi, i.e. l'outil réagit.
2. transformé, par le langage d'interaction en entrée, en tâche élémentaire qui s'applique à l'objet de la tâche.

L'objet de la tâche voit donc son état interne modifié par le langage d'interaction en entrée : ses propriétés numériques sont modifiées. Ce nouvel état interne est :

1. rendu observable grâce à la modalité de liaison en sortie. Des propriétés physiques de l'objet sont mises à jour elles aussi : c'est le retour d'information.
2. transformé par le langage d'interaction en sortie : c'est la réponse de l'objet de la tâche vers l'outil, qui peut participer à son tour à la réaction de l'outil.

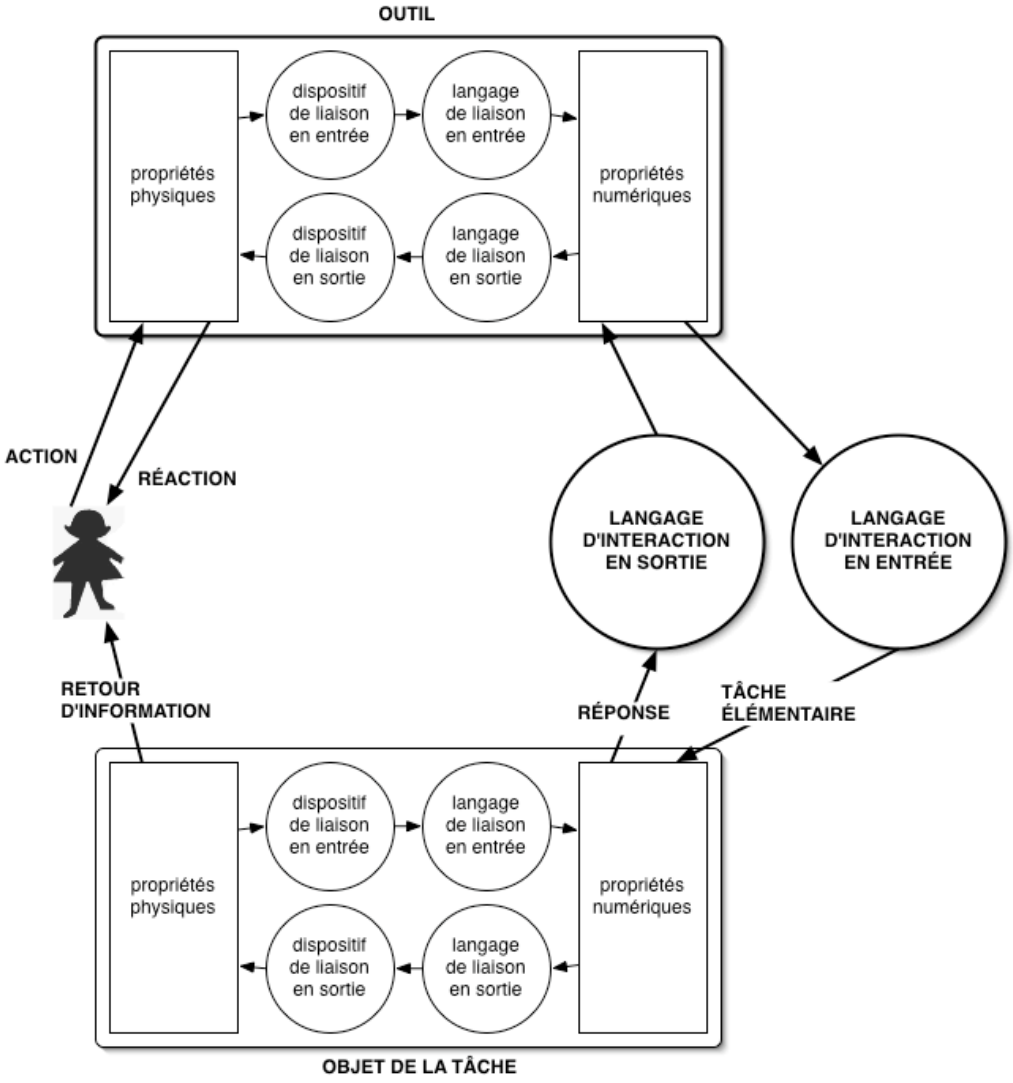


Figure 56 : Description générique d'une interaction simple à un bas niveau d'abstraction grâce au modèle d'interaction mixte, avec le détail des objets mis en jeu dans l'interaction.

Pour illustrer notre modèle d'interaction mixte, nous modélisons l'interaction dans le cas du DigitalDesk [Wellner, 1993] à la Figure 57. Dans le schéma, correspondant au scénario avec la gomme qui efface un dessin de maison à la fois dessiné au crayon et projeté par le système, l'interaction est décrite grâce au modèle d'interaction mixte.
L'utilisateur manipule la gomme. Ces mouvements, ainsi que leurs positions, sont reconnus par la modalité de liaison *(caméra, vision par ordinateur)*, pour ajuster les propriétés numériques de cet outil. Grâce au langage d'interaction, lorsque les mouvements reconnus sont ceux qui correspondent à l'action de « gommer », la tâche élémentaire ou commande est envoyée à l'objet de la tâche, le dessin mixte. Il n'y a pas de réaction de l'outil. La tâche élémentaire s'applique au dessin mixte, et le système efface certaines parties numériques du dessin qui se trouvent sous la gomme. L'utilisateur voit ce changement via le retour d'information : grâce à la modalité de liaison en sortie du dessin mixte, il voit que les parties qu'il a gommées ne sont plus projetées.

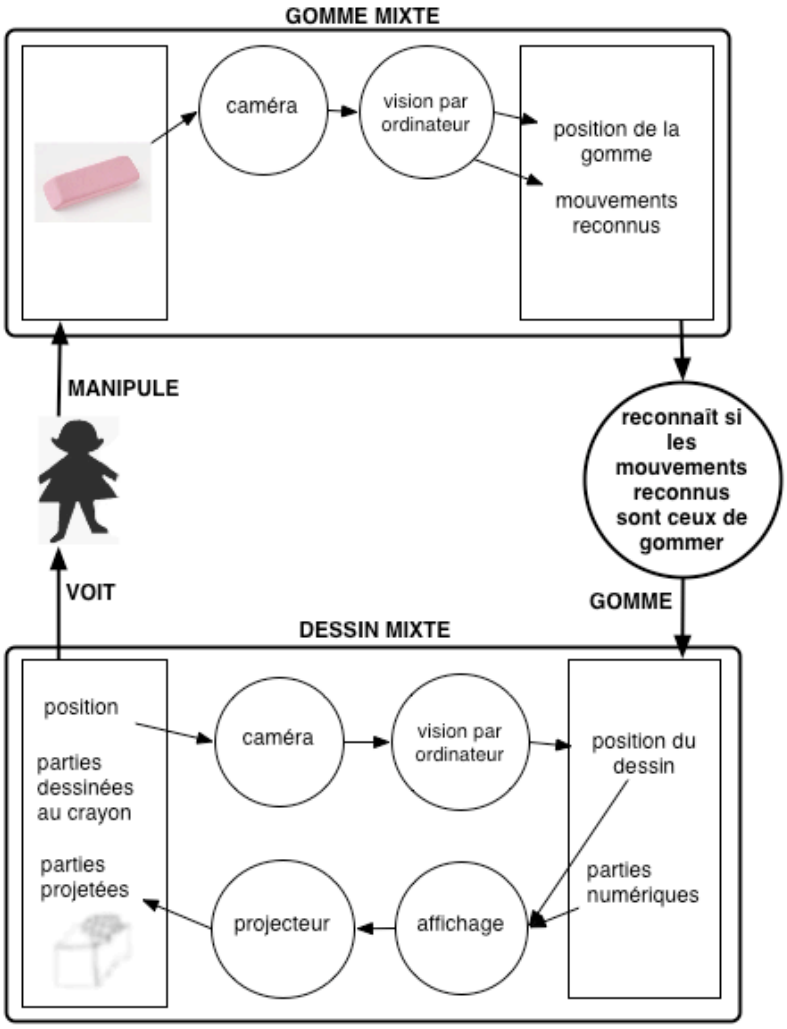


Figure 57 : Exemple du DigitalDesk décrit grâce au modèle d'interaction mixte : l'utilisateur gomme le dessin grâce à la gomme.

Préciser le langage d'interaction permet de décrire comment la gomme est utilisée pour gommer. Un concepteur pourrait concevoir un langage d'interaction alternatif à celui de la Figure 57, comme par exemple utiliser la position de la gomme (outil) pour gommer l'objet (dessin) situé sous l'outil. De plus, un langage d'interaction ne se confine pas aux modalités telles que la parole et permet également de décrire la manipulation directe : par exemple, il peut faire correspondre la position de l'outil avec la position de l'objet de la tâche.

1.3.2.2. Outil mixte et artéfact

Nous soulignons ici qu'une technique d'interaction avec un système de réalité mixte n'implique pas forcément la manipulation d'un artéfact. Nous avons décrit l'interaction avec les systèmes mixtes en termes de modalités d'interaction. Contrairement au modèle d'interaction instrumental initial qui focalise sur les instruments comme artéfacts, nous pouvons décrire l'interaction sans qu'il y ait nécessairement d'artéfact manipulé. Un microphone et le langage pseudo-naturel, ou la main, une caméra et la manipulation directe peuvent faire partie d'une modalité d'interaction avec un système de réalité mixte. Ainsi la notion d'outil d'interaction dépasse celle traditionnellement admise : le dispositif est soit un objet tangible manipulable, soit le dispositif d'une modalité d'interaction appartenant à un autre paradigme, comme la manipulation directe. Par exemple, avec le jeu *Tuttuki Bako* (Chapitre 2, section 2.2.2.2), l'utilisateur manipule directement l'objet de la tâche (un panda accroché à un fil à la Figure 58), en introduisant son doigt dans la boîte de jeu. C'est un cas de manipulation directe.

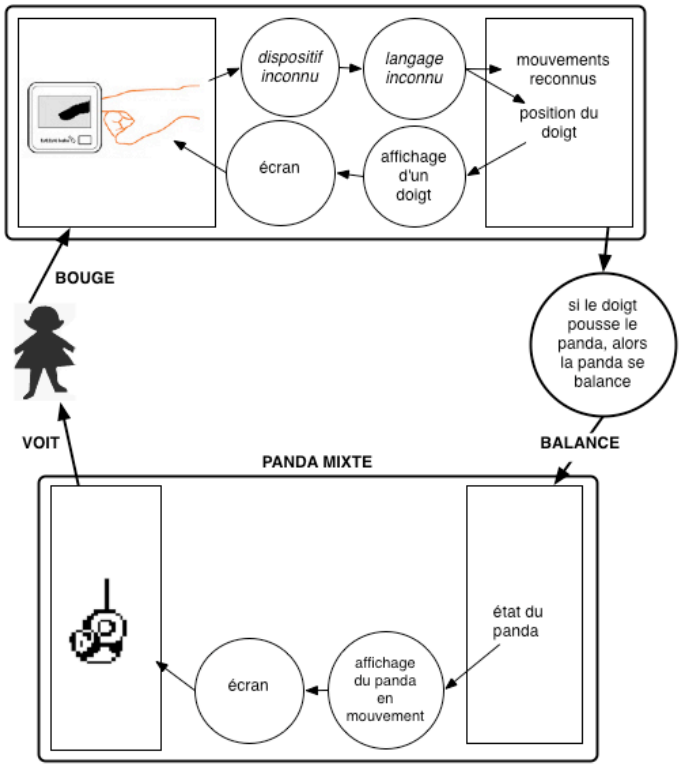


Figure 58 : Interaction avec le panda dans *Tuttiki Bako* (http://www.asovision.com/tuttuki/) : l'outil ne prend pas la forme d'un d'artefact, mais d'une modalité d'interaction directe. (Nous n'avons pas trouvé d'information sur la modalité en entrée pour le dispositif d'interaction.)

Ce cas, certes aux limites du modèle d'interaction instrumentale, mais néanmoins présent dans l'interaction avec les systèmes de réalité mixte, est donc intégré à notre modèle, car ce dernier est basé sur la notion de modalité d'interaction et non d'artéfact.

1.3.2.3. Interaction multimodale avec les systèmes de réalité mixte

L'interaction avec les objets peut elle aussi être multimodale. La composition des modalités d'interaction peut intervenir au niveau articulatoire (après les outils mixtes et dispositifs d'interaction), ou au niveau syntaxique et sémantique (après les langages d'interaction). La composition des modalités peut faire intervenir les différents aspects introduits dans [Vernier, Nigay, 2000]. La Figure 59 montre un exemple de description de l'interaction avec le modèle d'interaction mixte dans le cas d'une interaction bi-manuelle (donc multimodale) avec le système NavRNA [Bailly et al., 2006] (Chapitre 3, section 2.3) : l'utilisateur redimensionne la molécule d'ARN projetée sur la table, à l'aide de deux jetons.

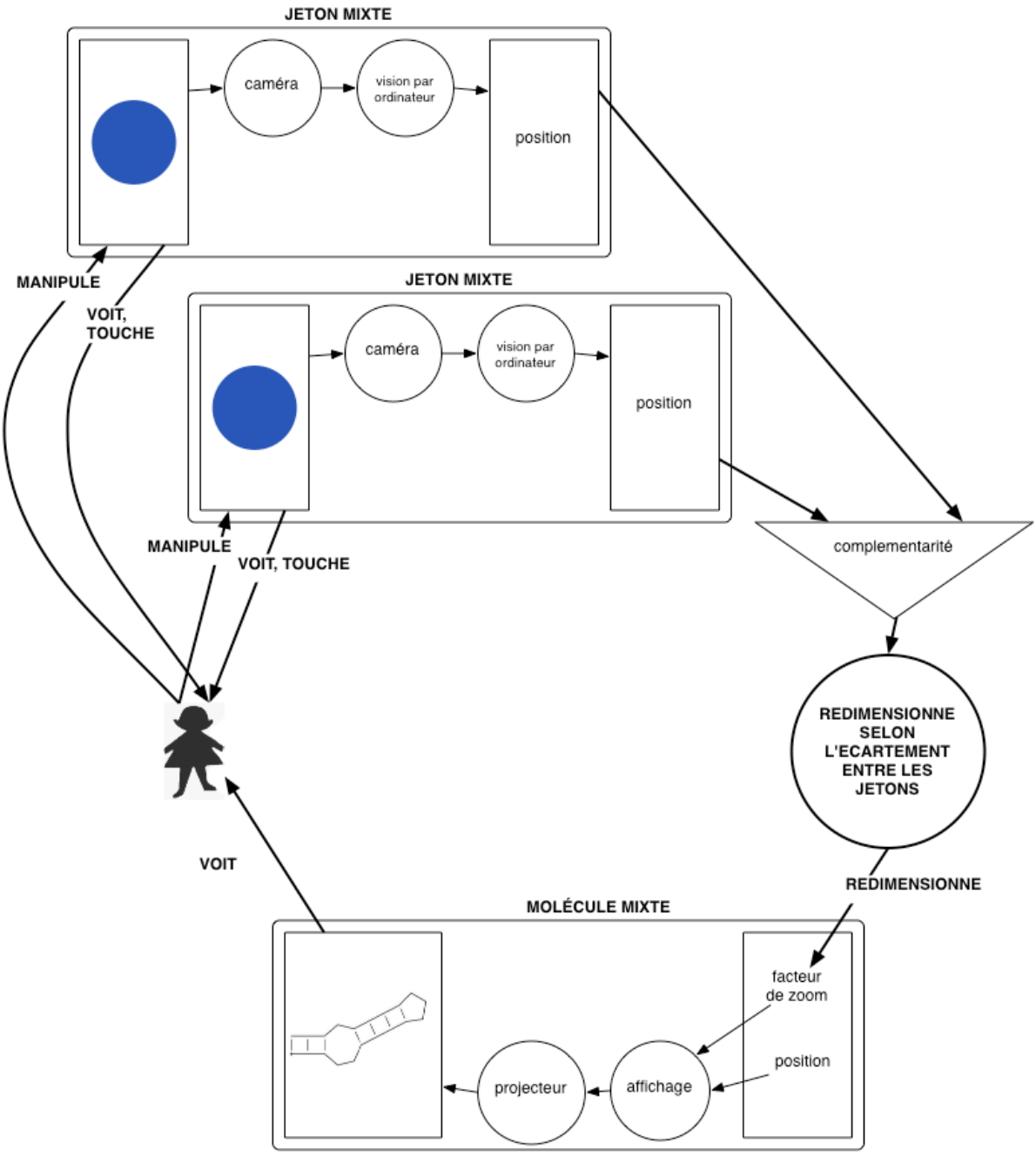


Figure 59 : Exemple d'interaction multimodale avec NavRNA [Bailly et al., 2006] : l'utilisateur manipule deux jetons pour redimensionner la molécule d'ARN projetée sur la table.

1.4. Conclusion sur la description de l'interaction centrée sur les objets

Nous avons introduit notre définition d'un objet mixte ou hybride, existant à la fois dans les mondes physique et numérique. Nous avons également décrit l'interaction avec de tels objets grâce au modèle d'interaction mixte. Nous avons vu que le modèle nous permettait de décrire l'interaction avec des systèmes très différents. Pour compléter notre démonstration du fort pouvoir descriptif de notre modèle, nous fournissons en Annexe A la description de l'interaction avec les multiples exemples, couvrant un éventail très large d'interfaces mixtes, introduits aux Chapitres 2 et 3.
Un modèle d'interaction à haut niveau d'abstraction a un pouvoir descriptif fort au détriment de son pouvoirs comparatif [Beaudouin-Lafon, 2004]. Un apport de notre modèle est de permettre plusieurs

niveaux d'abstraction dans la description (cf. Figure 55 et Figure 56). Cela permet donc de combiner les avantages d'un fort pouvoir descriptif avec un fort pouvoir comparatif. Nous focalisons maintenant sur le pouvoir comparatif du modèle, en considérant l'espace de caractérisation engendré.

2. Espace de caractérisation engendré

Nous présentons les caractéristiques liées au modèle d'interaction mixte. Ces caractéristiques permettent de comparer, classer les systèmes de réalité mixte. Elles définissent donc le pouvoir comparatif de notre modèle.
Au fil de la présentation, nous illustrons les caractéristiques en considérant plusieurs systèmes existants. Nous avons sciemment choisi des systèmes similaires et donc difficile à distinguer. Les systèmes ont déjà été introduits au Chapitre 3 section 2.3 pour justement illustrer les manques des modèles existants. Nous les reprenons ici pour appuyer notre démonstration du pouvoir comparatif de notre modèle. Nous considérons donc les sept systèmes de la section 2.3 du Chapitre 3, qui font tous intervenir la manipulation d'objets sur une table :

- Le DigitalDesk avec le bouton *FILL* de papier et la gomme mixtes,
- Le système PICO avec ses palets,
- Les jetons de NavRNA,
- Le Phicon en forme de dôme du MIT du Tangible Geospace,
- Les objets de la reacTable,
- Les music bottles,
- Les palets de l'actuated workbench.

Pour présenter l'espace de caractérisation du modèle d'interaction mixte, nous considérons d'abord les caractéristiques intrinsèques (niveau objet) des objets mixtes avec lesquels l'utilisateur interagit, pour ensuite présenter les caractéristiques extrinsèques de ces objets (niveau interaction).

2.1. Caractérisation intrinsèque

Les caractéristiques intrinsèques des objets mixtes sont celles qui sont propres à la façon dont est fait l'objet. Elles permettent donc de caractériser tour à tour les modalités de liaison, les propriétés physiques et les propriétés numériques.

2.1.1. Caractérisation des modalités de liaison

En nous appuyant sur des travaux en interaction multimodale pour définir le lien entre physique et numérique dans les systèmes mixtes, nous héritons également des taxonomies qui en découlent. Nous les rappelons ici, adaptées à la notion d'objet mixte.

2.1.1.1. Dispositif

La plupart des taxonomies des dispositifs d'interaction sont liées à la technologie et aux aspects matériels de l'informatique, et sont donc sujettes à ses fluctuations. Nous notons néanmoins une synthèse de ces taxonomies issues de [Buxton, 1983] et [Mackinlay et al., 1990] dans [Nigay, Coutaz, 1996] dont nous retenons ici les caractéristiques suivantes :

- Les caractéristiques du sens :
  - **Sens humain** (comme le toucher)
  - **Sens** ou **direction** (capteur ou effecteur, i.e. entrée ou sortie)
- Les caractéristiques temporelles :
  - **Temporalité** (éphémère ou non)
  - **Persistance**
- Les caractéristiques de l'information :
  - **Grandeur captée** (comme la pression, la température), que nous étendons à la **grandeur générée** pour le cas d'un dispositif de sortie

  - **Nombre de dimensions** (par exemple deux dimensions captées par la souris)
  - **Débit d'information** ou **fréquence**, caractéristique de précision temporelle
  - **Résolution**, caractéristique de précision spatiale
- Les caractéristiques spatiales :
  - **Spatialité** (lieu d'interaction comme l'écran ou la surface où le projecteur projette)
  - **Position** (lieu où le dispositif se situe). C'est une caractéristique identifiée dans [Mackay, 1996] que nous ajoutons, en plus de la spatialité d'un dispositif qui caractérise le lieu d'interaction.

À titre illustratif, nous montrons que l'intérêt de ces caractéristiques pour comparer des systèmes existants (pouvoir comparatif du modèle). Ainsi nous considérons les *music bottles* de la Figure 60 (trois projecteurs) et l'*actuated workbench* de la Figure 61 (un projecteur). Ces systèmes n'ont pas la même résolution : l'un affiche une tache de couleur sur et autour de la bouteille, alors que l'autre affiche une tache de couleur sur quelques millimètres carres à l'endroit précis où se trouve l'objet.



Figure 60 : Projection arbitraire à basse résolution pour les Music Bottles [Ishii et al., 2001].

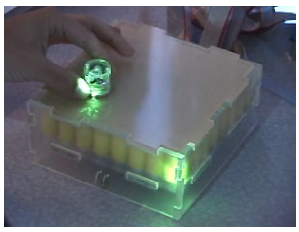


Figure 61 : Projection d'un point vert à haute résolution pour l'Actuated Workbench.

Capitaliser les travaux issus de la multimodalité nous permet de considérer des caractéristiques identifiées pour les systèmes de réalité mixte, comme ceux de [Dubois, 2001] [Dubois, Gray, 2007], pour décrire un composant adaptateur : indice entrée/sortie, son nombre de dimensions, le mécanisme de capture. Nous avons intégré explicitement la caractérisation de la position du dispositif de [Mackay, 1996]. De plus nous intégrons les caractéristiques de [Fitzmaurice et al., 1995] pour l'axe « entrée et sortie » qui correspondent à la grandeur captée et au nombre de dimensions. Nous intégrons aussi les « medium » (grandeur captée ou générée), « nimbus » (spatialité dans le cas d'un dispositif de sortie), « focus » (spatialité dans le cas d'un dispositif d'entrée) et « adaptateurs » (dispositif) de [Benford, Fahlen, 1993], ainsi que les « mouvements capturés » (grandeur captée) et leurs caractéristiques (degré de liberté, précision, etc.) dans [Benford et al., 2005].

2.1.1.2. Langage

Pour caractériser le langage d'une modalité de liaison, nous retenons le travail de [Bernsen, 1993] sur l'interaction multimodale. Il identifie un ensemble de propriétés s'appliquant au langage d'une modalité. Les langages peuvent être de nature :

- **Statique** ou **dynamique** : Le caractère dynamique ou statique d'une modalité repose sur la présence de la dimension temporelle dans la représentation. Par exemple, la modalité de sortie pour les objets de la reacTable (Figure 62) est dynamique car elle affiche des animations autour des objets qui sont fonctions du temps.

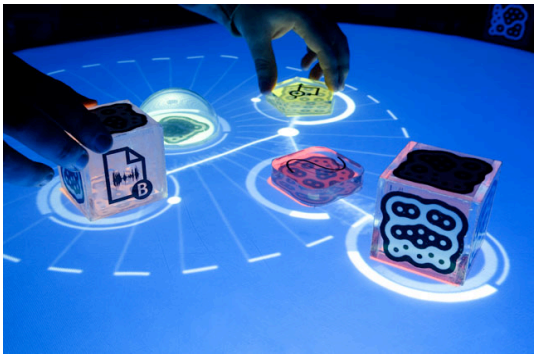


Figure 62 : ReacTable [Jordà et al., 2007].

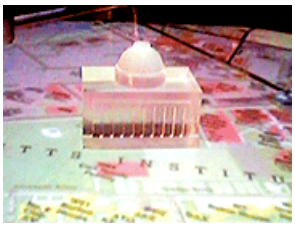


Figure 63 : Tangible geospace [Ullmer, Ishii, 1997], avec deux objets mixtes : le Phicon et la carte.

- **Linguistique** ou non : Une modalité linguistique s'appuie sur un langage, c'est-à-dire un système structuré de signes remplissant une fonction de communication. Par exemple, le langage utilisé pour représenter la carte du Tangible geospace (Figure 63) est en partie linguistique : une partie de la représentation est sous forme de langage naturel écrit, comme le nom des bâtiments. Au contraire, le langage utilisé pour représenter la position de l'objet sur l'*actuated workbench* (le point vert de la Figure 61), n'est pas linguistique.
- **Analogique** ou non : Une modalité analogique entretient un rapport de ressemblance avec la réalité : elle fonctionne par ressemblance à l'élément signifié. Par exemple, le langage utilisé pour représenter la carte du Tangible geospace (Figure 63) est en parti analogique : le dessin du plan est une représentation analogique par rapport au terrain qu'il représente, vu de dessus.
- **Arbitraire** ou non : Un langage arbitraire fonctionne en dehors d'un système conventionnel et ne présente aucun indice sémantique sur le signifié. C'est l'exemple du langage de sortie des *music bottles* (Figure 60) : les couleurs ne donnent aucun indice sur la hauteur et le volume du son. À l'inverse, les langages de liaison non arbitraires reposent, pour remplir leur rôle, sur l'existence d'un système sémantique appris. C'est par exemple la représentation en croix de la position de l'aiguille de ponction dans CASPER.

À ces caractéristiques s'ajoutent celles qui sont proposées dans [Vernier, Nigay, 2000]. Un langage peut être :

- **Partiel** ou **global**, selon que le langage de liaison transmet seulement une partie ou la totalité de l'information. Cette caractéristique permet de faire la différence entre les deux langages de liaison en sortie des *music bottles* : le langage de la modalité sonore est global, puisque toute l'information sur la musique numérique est perceptible, alors que le langage visuel à base de couleur (Figure 60) ne rend compte que du volume et de la hauteur du son.
- **Précis** ou **vague** : Le langage de liaison peut varier en précision. Par exemple, pas besoin d'identifier la position précise des *music bottles* sur la table (Figure 60), un langage de liaison en entrée qui soit vague suffira pour savoir dans lequel des trois parties elle se trouve. Au contraire, pour le bouton *FILL* du DigitalDesk, il est nécessaire de considérer langage de liaison en entrée qui soit précis.
- **Déformant** ou non : la syntaxe d'un langage de liaison peut être déformante, comme celle du langage de liaison en entrée de la tackball qui traduit le deplacement de la boule en un déplacement en deux dimensions grâce à une homothétie (un petit déplacement de la boule permet un grand déplacement sur la surface). Un langage de liaison peut être non

déformant, comme le langage de liaison en entrée des jetons de NavRNA, dont le déplacement physique correspond exactement au déplacement numérique.

À travers les exemples utilisés pour présenter ces caractéristiques des langages de liaison, nous avons constaté qu'elles permettent de faire la différence entre les objets mixtes mis en jeu dans l'interaction. De plus, grâce à ces caractéristiques, non seulement nous englobons les travaux issus de l'interaction multimodale, mais, et c'est le plus important, ceci nous permet de capitaliser des caractéristiques déjà identifiées pour les interfaces mixtes, comme le type de langage [Dubois, 2001] ou la représentation [Dubois, Gray, 2007].

2.1.1.3. Degré d'intégration

Le degré d'intégration [Beaudouin-Lafon, 2004] est le nombre de degrés de liberté du dispositif d'entrée qui sont utilisés par l'instrument. Pour les interfaces mixtes, cette caractéristique a été aussi évoquée dans [Dubois, Gray, 2007, section 3.4.2.1.].
Nous généralisons cette caractéristique à l'objet mixte en général, qu'il soit outil d'interaction ou non, et nous considérons le degré d'intégration d'un objet en entrée mais également en sortie. Dans notre description d'un objet mixte, il s'agit d'étudier la relation entre les caractéristiques du dispositif et du langage de liaison, que ce soit en entrée ou en sortie. Par exemple, les *music bottles* ont un fort degré d'intégration en sortie, malgré la basse résolution des dispositifs de projection (projecteurs de lumière) : il n'y a pas de précision dans le langage qui serait perdue par le dispositif de sortie.

2.1.1.4. Liaisons multimodales

La composition des modalités de liaison peut être caractérisée selon trois axes indépendants (Tableau 4) :
Les **aspects** : temporel et spatial,

1. Les **niveaux d'abstraction** : articulatoire (dispositif) et syntaxique (langage),
2. Leur composition **sémantique**.

La composition peut être caractérisée selon cinq schémas de composition de [Vernier, Nigay, 2000] issus des relations d'Allen. Le cadre de caractérisation de la composition des modalités de liaison qui en découle est présenté au Tableau 4.

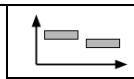
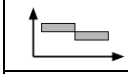
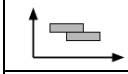
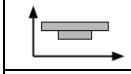
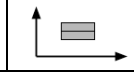
		Aspects		*Niveaux d'abstraction*		
		Temporel	*Spatial*	*Articulatoire*	*Syntaxique*	*Sémantique*
Schémas		Anachronisme	Disjonction	Indépendance	Différence	Concurrence
		Séquence	Adjacence	Fusion/Fission	Complétion	Complémentarité
		Concomitance	Intersection	Fusion/Fission et Duplication	Divergence	Complémentarité et redondance
		Coïncidence	Imbrication	Duplication partielle	Extension	Redondance partielle
		Parallélisme	Recouvrement	Duplication totale	Gémellité	Redondance totale

Tableau 4 : Caractérisation de la composition des modalités dans une liaison multimodale, d'après [Vernier, Nigay, 2000].

Par exemple à la Figure 52 page 77, l'aiguille de ponction utilisée dans CASPER [Dubois, 2001] a une liaison multimodale en entrée, dont la composition s'effectue au niveau articulatoire, avec une fusion des données (Tableau 5, en gris). Nous caractérisons cette composition des modalités de liaison par un parallélisme temporel et un recouvrement spatial, ainsi qu'une complémentarité sémantique (Tableau 5, en orange).

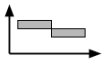
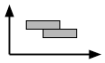
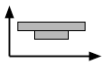
	Aspects		*Niveaux d'abstraction*		
	Temporel	*Spatial*	*Articulatoire*	*Syntaxique*	*Sémantique*
	Anachronisme	Disjonction	Indépendance	Différence	Concurrence
	Séquence	Adjacence	Fusion/Fission	Complétion	Complémentarité
	Concomitance	Intersection	Fusion/Fission et Duplication	Divergence	Complémentarité et redondance
	Coïncidence	Imbrication	Duplication partielle	Extension	Redondance partielle
	Parallélisme	Recouvrement	Duplication totale	Gémellité	Redondance totale

Tableau 5 : Caractérisation de la composition des modalités pour l'aiguille de ponction de CASPER.

Au contraire, dans l'exemple des *music bottles* de la Figure 53 page 77, la décomposition s'effectue au niveau syntaxique. La modalité sonore est l'extension syntaxique de l'autre (Tableau 6, en gris). Elles sont aussi temporellement parallèles, spatialement imbriquées, et sémantiquement partiellement redondantes (Tableau 6, en orange).

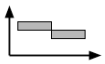
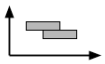
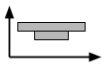
	Aspects		*Niveaux d'abstraction*		
	Temporel	*Spatial*	*Articulatoire*	*Syntaxique*	*Sémantique*
	Anachronisme	Disjonction	Indépendance	Différence	Concurrence
	Séquence	Adjacence	Fusion/Fission	Complétion	Complémentarité
	Concomitance	Intersection	Fusion/Fission et Duplication	Divergence	Complémentarité et redondance
	Coïncidence	Imbrication	Duplication partielle	Extension	Redondance partielle
	Parallélisme	Recouvrement	Duplication totale	Gémellité	Redondance totale

Tableau 6 : Caractérisation de la composition des modalités pour les *music bottles*.

2.1.1.5. Couplage des modalités de liaison en entrée et en sortie

Nous avons caractérisé les modalités composées grâce au cadre de conception présenté au Tableau 4. Nous exploitons aussi ce cadre de conception pour caractériser les relations entre modalités de liaison en entrée et celles en sorties. Nous caractérisons par exemple la composition entre les liaisons entrée/sortie du palet de l'*actuated workbench* utilisé avec la vision par ordinateur : l'entrée et la sortie ne sont parfois pas parallèles (aspect temporel du Tableau 4), mais plutôt en séquence, car décalées à cause de la latence logicielle, comme l'illustre la Figure 64.

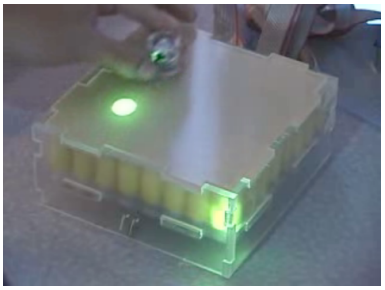


Figure 64 : Du retard dans la liaison de l'objet manipulé avec l'*actuated workbench*.

En étudiant la composition des modalités d'entrée et de sortie d'un objet mixte, nous définissons la compatibilité et continuité perceptuelles et cognitives [Dubois, 2001] d'un objet. Ces deux propriétés prennent tout leur sens dans le contexte de notre modèle, en explicitant les deux niveaux perceptuel et cognitif qui correspondent respectivement aux niveaux dispositif et langage de la modalité de liaison.
Ces propriétés générales ont été plus étudiées dans le cas de la compatibilité et continuité spatiales. Nous constatons que la forme de l'incarnation dans [Fishkin, 2004] ainsi que la composition spatiale entre éléments physiques et numériques de [Fitzmaurice et al., 1995] sont explicites dans notre espace de caractérisation en considérant l'aspect spatial (Tableau 4) de la composition des modalités de liaison.

En synthèse, en exploitant les résultats issus de l'interaction multimodale au cas des modalités de liaison d'un objet, nous généralisons et étendons des caractéristiques existantes des interfaces mixtes :

- ASUR [Dubois, 2001] et son adaptateur avec son indice entrée/sortie, et ses caractéristiques (nombre de dimensions, type de langage [Dubois, 2001], mécanisme de capture, représentation [Dubois, Gray, 2007]). Nous retrouvons ces concepts grâce à la notion de modalité de liaison, son dispositif et son langage,
- L'axe « entrée et sortie » de [Fitzmaurice et al., 1995] qui caractérise le dispositif,
- La taxonomie de [Mackay, 1996] qui caractérise le dispositif,
- Les « medium », « nimbus », « focus » et « adaptateurs » de [Benford, Fahlen, 1993], qui caractérisent aussi les dispositifs et leurs relations spatiales,
- Les « mouvements capturés » et leurs caractéristiques (degré de liberté, précision, etc.) [Benford et al., 2005] qui s'appliquent aussi directement aux modalités de liaison.

Ces caractéristiques existantes, outre le fait qu'elles soient explicites dans notre modèle, sont étendues :

- Pour la composition spatiale, nous définissons cinq schémas là où [Fitzmaurice et al., 1995] n'en voyait que deux ou [Fishkin, 2004] quatre pour l'incarnation.
- Pour la composition temporelle, nous définissons cinq schémas pour caractériser la relation entre la modalité d'entrée et la modalité de sortie, là où [Fitzmaurice et al., 1995] n'en voyait que deux (temps réel/différé) : Les modalités de liaison d'un objet mixte peuvent être asynchrone, en séquence, concomitante, coïncidente, ou parallèle.

2.1.2. Caractérisation des propriétés physiques

2.1.2.1. Affordance des propriétés physiques

Les propriétés physiques d'un objet mixte peuvent être caractérisées par leur affordance. L'affordance est définie dans [Norman, 1999] comme l'aspect d'un objet qui suggère comment cet objet peut être utilisé. Par exemple, un objet plat peut être glissé le long d'une surface plane, comme le Phicon du Tangible geospace ou les objets de la reacTable. Si l'on considère la symétrie de rotation parmi les exemples, nous identifions alors deux classes d'objets :

1. Ceux qui ont une symétrie de rotation, que la rotation n'affecte pas : les jetons de NavRNA, les palets de l'*actuated workbench* et de PICO.

2. Ceux qui n'ont pas de symétrie de rotation : le Phicon du Tangible geospace, les objets de la reacTable, les *music bottles*, et le bouton et la gomme du DigitalDesk. Nous pouvons supposer que ces objets seront tournés plus fréquemment que les objets de la première classe.

Considérer l'affordance d'un objet mixte est à mettre en relation avec la notion de « mouvements attendus » [Benford et al., 2005].

2.1.2.2. Propriétés physiques capturées et générées

Nous considérons les propriétés physiques indépendamment des modalités de liaison. Quelle que soit la modalité choisie, les propriétés physiques peuvent être captées en entrée, et/ou générées en sortie : il en découle un espace des possibilités structuré en quatre zones (Figure 65) :

1. Les propriétés physiques ni captées ni générées (en gris à la Figure 65),
2. Les propriétés physiques captées mais pas générées (en bleu à la Figure 65),
3. Les propriétés physiques captées et générées (en violet à la Figure 65),
4. Les propriétés physiques non captées mais générées (en rouge à la Figure 65).

Cette caractéristique étend des notions déjà existantes comme celles de l'axe entrée-sortie de [Fitzmaurice et al., 1995] ou la notion de « mouvements captés » [Benford et al., 2005].

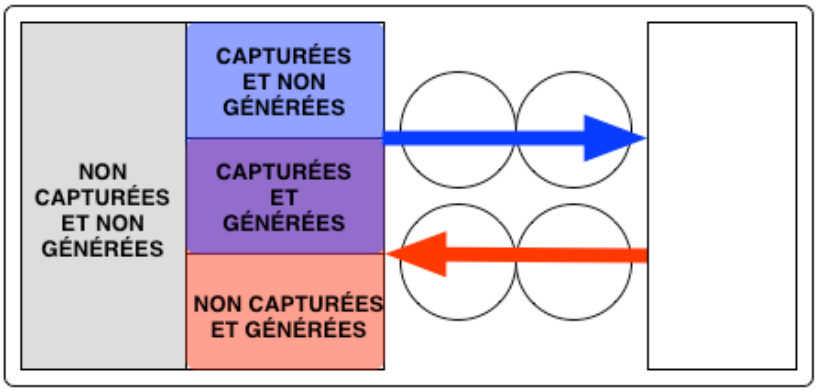


Figure 65 : Caractérisation intrinsèque des propriétés physiques d'un objet mixte selon leur prise en compte par une modalité de liaison en entrée ou en sortie.

Par exemple, le jeton bleu de NavRNA (Figure 66) a, parmi ses propriétés physiques, la position sur la table et la couleur. La position sur la table est clairement capturée, mais il convient de ne pas oublier que la couleur l'est aussi, même si elle n'est qu'un moyen pour trouver la position de l'objet. La caractériser comme propriété physique capturée permet d'expliciter le fait que la couleur ne doit pas être modifiée pour le concepteur lors d'une refonte de l'aspect physique d'un jeton. Cette remarque nous conduit à identifier deux types de propriétés capturées : celles qui le sont pour résoudre un problème technologique (« transparentes » pour l'utilisateur mais pas pour le concepteur), et celles qui le sont pour l'interaction.

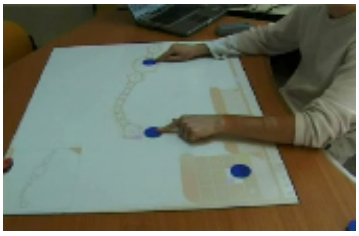


Figure 66 : NavRNA [Bailly et al., 2006].

Considérons maintenant la position physique des objets des sept systèmes exemples. Nous obtenons la classification présentée au Tableau 7. Tous ces objets partagent une même propriété physique : ils ont une position physique. Pour tous, la position est captée, sauf pour le palet de l'*actuated*

workbench lorsqu'il n'est pas en mode vision par ordinateur (Figure 67). Cette position physique est générée pour seulement deux systèmes : PICO et l'*actuated workbench*, où les palets sont déplacés par le système par des aimants. Nous obtenons donc la classification en trois ensembles présentée au Tableau 7.

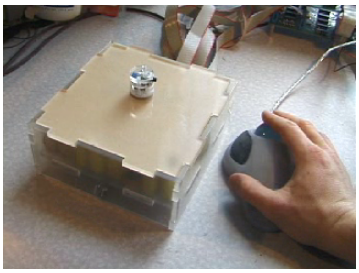


Figure 67 : *Actuated workbench* utilisé avec une souris : la position du palet n'est pas capturée par une caméra.

Position physique	**Générée**	**Non Générée**
Capturée	PICO Actuated workbench (utilisé avec la vision par ordinateur)	NavRNA Tangible Geospace reacTable Music bottles Digital Desk
Non Capturée	Actuated workbench (utilisé avec la trackball)	

Tableau 7 : Caractérisation de la position physique de l'objet mixte manipulé sur une table.

Pour affiner cette classification, nous considérons maintenant une autre propriété physique : la couleur (Tableau 8). Pour les objets de NavRNA et la reacTable, la couleur est capturée par la modalité de liaison en entrée afin de calculer la position de l'objet. Pour la reacTable, la modalité de liaison consiste à suivre les marqueurs représentés sur les cubes, si bien que leurs couleurs ne peuvent pas être modifiées, ou d'une façon très limitée. De la même façon pour NavRNA, la couleur ne peut pas être modifiée. Au contraire, pour les *music Bottles*, PICO ou l'*actuated workbench*, la couleur n'est pas capturée : la modalité ne capte que la lumière infrarouge émise par les objets dans le cas de PICO et de l'*actuated workbench*, et un marqueur électromagnétique est utilisé pour les *music bottles*. Par contre une couleur est générée pour ces trois systèmes. Enfin le dernier cas est celui du Phicon du Tangible Geospace pour lequel la couleur n'est ni générée, ni capturée (le rayonnement infrarouge est utilisé). Nous n'incluons pas le DigitalDesk dans cette classification car le système n'a jamais été complètement développé : en effet, lors du développement, des contraintes technologiques peuvent avoir un impact direct sur les propriétés physiques capturées et/ou générées comme dans NavRNA. La classification obtenue est présentée dans le Tableau 8.

Couleur	**Générée**	**Non Générée**
Capturée		NavRNA reacTable
Non Capturée	Music bottles PICO Actuated workbench (utilisé avec la vision par ordinateur)	Tangible Geospace Actuated workbench (utilisé avec la trackball)

Tableau 8 : Caractérisation de la couleur de l'objet mixte manipulé sur une table.

2.1.2.3. Propriétés physiques à rebond

Nous considérons la caractéristique de rebond identifiée dans [Dix et al., 2007]. Une propriété physique peut être « à rebond » ou non. Parmi nos exemples, seule la position physique des objets de PICO est à rebond, comme l'illustre la Figure 68 : Ils ont une position d'équilibre vers laquelle

ils reviennent si l'utilisateur les déplace. Au contraire, dans les autres exemples, comme pour les jetons de NavRNA, la position physique n'est pas à rebond.

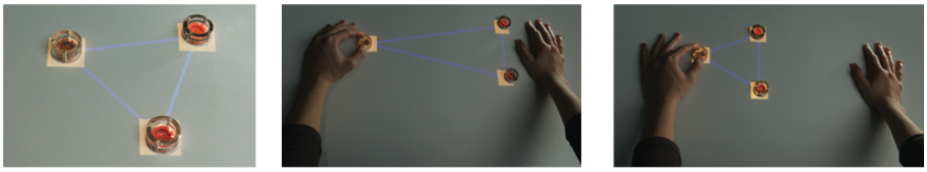


Figure 68: Les objets de PICO ont une position physique d'équilibre. Illustration extraite de http://www.jamespatten.com/pico/picovid.html.

La caractéristique d'une propriété à rebond n'est pas forcément définie par les modalités de liaisons, comme dans l'exemple de PICO. Ainsi un palet attaché à un élastique physique fixé à la table aurait une position physique à rebond, car il reviendrait à sa position d'équilibre.

2.1.2.4. Spatialité et temporalité des propriétés physiques

En étudiant le couplage spatio-temporel des modalités de liaison (section 2.1.1.5) nous avons identifié des aspects de compatibilité et continuité d'un objet mixte conçu. Pour cela nous avons considéré que les caractéristiques des propriétés physiques capturées (modalités de liaison en entrée) découlaient des caractéristiques des modalités de liaison en entrée et en sortie mises en jeu. Or, toute propriété physique n'est pas capturée ou générée. Dans ce contexte, il convient donc d'étudier aussi le couplage spatio-temporel entre des propriétés capturées/générées (issus de modalités de liaison) et des propriétés non-capturées/non-générées.
Les propriétés physiques capturées/générées (issus de modalités de liaison) et des propriétés non-capturées/non-générées peuvent être asynchrones, en séquence, concomitantes, coïncidentes, ou en parallèle (aspect temporel, Tableau 4), et elles peuvent être séparées, adjacentes, s'intersecter, être imbriquée, ou se recouvrir (aspect spatial, Tableau 4).
Par exemple, la projection des cubes de la reacTable est adjacente à la partie non générée de l'objet (Figure 62), alors que pour l'*actuated workbench* utilisé avec la modalité (camera, vision par ordinateur), la projection recouvre la partie non générée de l'objet (Figure 61).
Grâce à cette caractérisation, nous capitalisons et affinons la compatibilité et continuité perceptuelle (au niveau des propriétés physiques) de [Dubois, 2001]. Il s'agit également d'un aspect (au niveau des propriétés physiques) de l'incarnation de [Fishkin, 2004] et de la composition spatiale entre éléments physiques et numériques de [Fitzmaurice et al., 1995].

2.1.3. Caractérisation des propriétés numériques

2.1.3.1. Propriétés numériques acquises et perceptibles

Symétriquement au côté physique, nous considérons les propriétés numériques indépendamment des modalités de liaison. Quelle que soit la modalité choisie, les propriétés numériques peuvent être acquises à partir du monde physique, via la modalité de liaison en entrée et/ou perceptibles dans le monde physique, via la modalité de liaison en sortie: il en découle un espace des possibilités structuré en quatre zones (Figure 69).

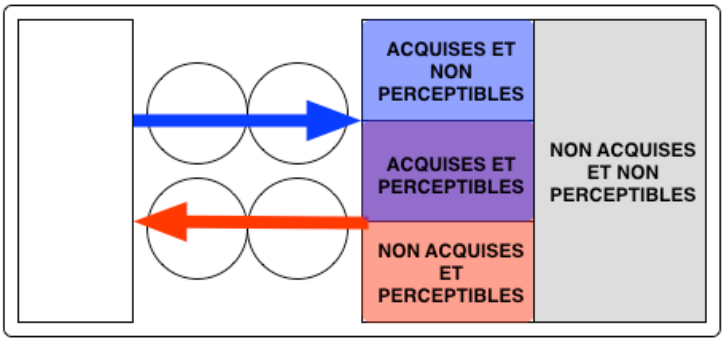


Figure 69 : Caractérisation intrinsèque des propriétés numériques d'un objet mixte selon leur prise en compte par une modalité de liaison en entrée ou en sortie.

Considérons par exemple la propriété numérique qui représente la position d'un objet sur une table. Pour la plupart de nos exemples, cette propriété est un couple de coordonnées (x, y). Les *music bottles* sont les seuls objets qui n'ont pas besoin d'une telle précision pour leur position numérique. Le système n'a besoin que de connaître la zone est où est la bouteille, parmi les trois qui partitionnent la table.
Nous considérons d'abord le caractère acquis ou non de la position numérique. Pour les sept exemples considérés, la position numérique est acquise à partir de la position des objets sur la table, sauf pour le cas de l'*actuated workbench* utilisé avec la trackball (Figure 70). En effet, dans le cas de l'*actuated workbench*, la position numérique de l'objet n'est pas acquise : elle est mise à jour de façon indirecte via un outil (trackball) et l'objet n'a pas de modalité de liaison en entrée pour acquérir cette propriété numérique (Figure 71).

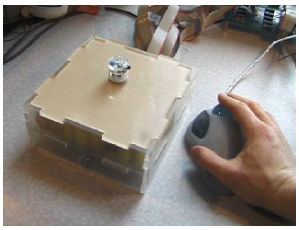


Figure 70 : L'*actuated workbench* utilisé avec la trackball.

Nous considérons maintenant le caractère perceptible de la position numérique. Pour l'*actuated workbench*, la position numérique est perceptible via la table aimantée. De plus, pour la reacTable, l'*actuated workbench* utilisé avec la vision par ordinateur, les *music bottles* et PICO, la position numérique est aussi perceptible/matérialisée, via une projection sur la table. En revanche pour le jeton de NavRNA, la position numérique n'est pas matérialisée. Il n'y a donc pas d'observabilité de son état interne. La classification obtenue est présentée au Tableau 9.

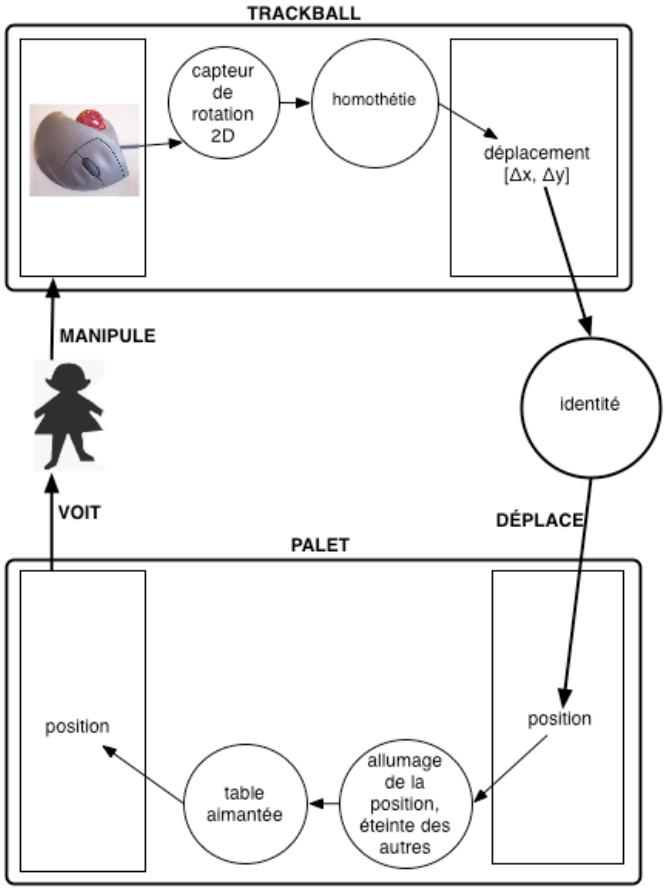


Figure 71 : Description de l'interaction avec l'*actuated workbench*, à l'aide d'une trackball.

Position numérique	**Perceptible**	**Non Perceptible**
Acquise	Actuated workbench (utilisé avec la vision par ordinateur) reacTable Music bottles PICO	NAVRNA Tangible Geospace Digital Desk
Non Acquise	Actuated workbench (utilisé avec la trackball)	

Tableau 9 : Caractérisation de la position numérique de l'objet manipulé sur la table.

2.1.3.2. Propriétés numériques à rebond

Comme pour les propriétés physiques, nous considérons aussi que des propriétés numériques peuvent être à rebond. Les propriétés numériques peuvent se comporter comme un ressort lorsqu'elles sont modifiées, et revenir à leur valeur initiale après un laps de temps fixé. Par exemple, la propriété numérique représentant la position des objets dans PICO est une propriété numérique à rebond. Par conséquent, la propriété physique générée à partir de la propriété numérique est aussi à rebond (section 2.1.2.3).

2.1.4. Synthèse

Nous avons présenté les caractéristiques intrinsèques d'un objet mixte. Dans ce cadre, nous avons vu que notre modélisation d'un objet mixte permettait de capitaliser des caractéristiques existantes pour les systèmes de réalité mixte, comme les mouvements ou actions attendus et captés [Benford et al., 2005] via l'affordance et les propriétés physiques captées. Nous explicitons aussi dans notre

modèle les propriétés physiques ou numériques à rebond [Dix et al., 2007], ainsi que la compatibilité et continuité spatiale de [Dubois, 2001] en considérant des aspects de composition des propriétés et modalités de liaison. Nous intégrons également les axes entrée-sortie et la temporalité du lien de [Fitzmaurice et al., 1995] et une forme de l'incarnation de [Fishin, 2004].
Nous capitalisons également des caractéristiques venant d'autres domaines de recherche que celui des interfaces mixtes, comme l'affordance [Norman, 1999], les caractéristiques des dispositifs [Buxton, 1983] [Mackinlay et al., 1990] et des langages [Bernsen, 1993] [Vernier, Nigay, 2000], la composition des modalités [Vernier, Nigay, 2000] et le degré d'intégration [Beaudouin-Lafon, 2000, 2004].
Notre description d'un objet mixte nous a également conduit naturellement à identifier de nouvelles caractéristiques, comme les propriétés physiques générées, les propriétés numériques acquises et matérialisées, les propriétés numériques à rebond, et la composition des propriétés physiques en cinq schémas.
Ceci met clairement en exergue notre contribution : non seulement le modèle d'interaction mixte permet d'unifier les approches existantes et d'organiser les caractéristiques dans un cadre cohérent, mais il nous permet d'identifier de nouvelles caractéristiques.
Nous présentons maintenant notre contribution pour les caractéristiques extrinsèques (niveau interaction), comme nous venons de le faire pour les caractéristiques intrinsèques (niveau objet).

2.2. Caractérisation extrinsèque

Les caractéristiques extrinsèques des objets mixtes sont celles qui dépendent des circonstances extérieures. Elles permettent de caractériser le rôle de l'objet, les langages d'interaction associés à l'objet, les relations entre objets, la métaphore appliquée à l'objet et ses ports d'entrée/sortie.

2.2.1. Caractérisation des rôles

Dans le modèle d'interaction mixte, le rôle de l'objet dans l'interaction est explicité : un objet mixte peut être outil, dispositif d'une modalité d'interaction, ou objet de la tâche, qui est le focus de l'attention de l'utilisateur. Par exemple, le bouton *FILL* (Figure 7) et la gomme du DigitalDesk (Figure 8) sont des outils, alors que le dessin de la maison est l'objet de la tâche. Les objets de la reacTable sont des outils alors que le son produit est l'objet de la tâche. Les jetons de NavRNA sont des outils, et l'objet de la tâche est la molécule d'ARN. De même, le Phicon du Tangible Geospace est un outil, alors que l'objet de la tâche est la carte.



Figure 72 : Remplir le toit de tuiles avec le DigitalDesk.

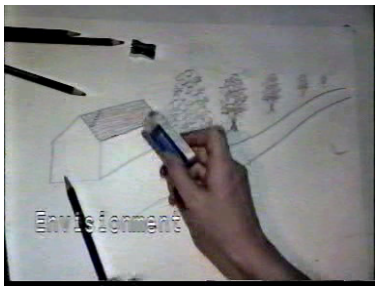


Figure 73 : Gommer une partie des tuiles avec le DigitalDesk.

Les deux rôles d'un objet mixte correspondent aux concepts d'instrument et d'objet du domaine de [Beaudouin-Lafon, 2004], mais aussi aux indices « outil» et « objet» d'un composant ASUR [Dubois, 2001], et aux éléments O et T d'IRVO [Chalon, 2004]. L'outil correspond à un objet verbe [Underkoffler, Ishii, 1999] et aux outils de [Holmquist et al., 1999].

2.2.2. Caractérisation des langages d'interaction associés

Le langage d'interaction associé à l'outil mixte peut être caractérisé tout comme un langage de liaison. Ainsi il peut être statique ou dynamique, linguistique ou non, analogique ou non, arbitraire ou non, partiel ou global, précis ou vague, déformant ou non. Par exemple, le langage d'interaction utilisé avec la reacTable n'exploite pas toutes les positions que l'utilisateur peut atteindre avec son outil : il s'agit donc d'un langage d'interaction partiel, au contraire du Tangible Geospace, où n'importe quelle position de l'outil sur la table aura un effet sur l'objet de la tâche.
Nous identifions ici une nouvelle caractéristique des systèmes de réalité mixte grâce à l'application des résultats en interaction multimodale.

2.2.3. Caractérisation de l'interaction multimodale

Dans le cas d'une interaction multimodale, comme pour redimensionner la molécule d'ARN dans NavRNA en manipulant deux jetons (Figure 59), nous caractérisons cette composition des modalités d'interaction selon les aspects et les schémas de composition présentés au Tableau 4. Nous rappelons que nous avions déjà utilisé ce résultat issu de l'interaction multimodale pour l'appliquer au cas des modalités de liaison (section 2.1.1.4). Cette composition des modalités peut avoir lieu au niveau des dispositifs d'interaction (Figure 59), ou au niveau des langages comme dans l'exemple du « Mets ça là » [Bolt, 1980] (Figure 74).

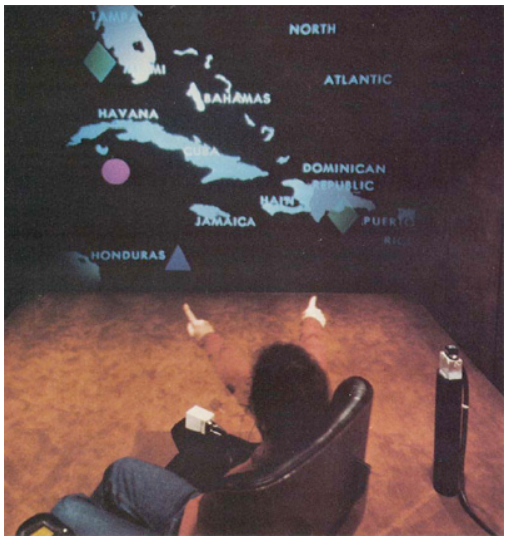


Figure 74 : « Mets ça là ». Illustration extraite de [Bolt, 1980].

Modéliser un objet mixte comme le dispositif d'une modalité d'interaction nous permet d'étendre des caractéristiques des systèmes de réalité mixte comme le nombre d'objets manipulés en parallèle [Fitzmaurice et al., 1995] en considérant les cas de multimodalité. Ainsi, en ne traitant que l'aspect temporel, nous ne considérons pas seulement l'utilisation de dispositifs d'interaction en parallèle, mais aussi leur utilisation anachronisme, en séquence, concomitante, coïncidente, et parallèle.

2.2.4. Caractérisation spatiale et temporelle des objets entre eux

Outre le cas de l'interaction multimodale mettant en jeu plusieurs objets mixte qui sont des dispositifs, nous étudions aussi les relations entre un objet mixte outil et un objet mixte objet de la tâche. Là encore, nous réutilisons les aspects temporels et spatiaux issus de la multimodalité que nous rappelons au Tableau 10.

Par exemple le bouton *FILL* et le dessin du DigitalDesk sont adjacents. La gomme, quant à elle, est imbriquée dans le plan horizontal du dessin et adjacente verticalement au dessin, tout comme le Phicon avec la carte, le bouchon avec la bouteille des *music bottles* et les jetons de NavRNA avec la molécule. Les outils de la reacTable sont complètement imbriqués dans la musique : ils sont perceptibles au moins là où est perceptible la musique (objet de la tâche).

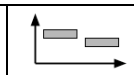
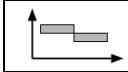
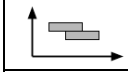
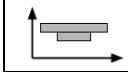
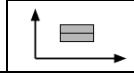
Schémas	*Temporel*	*Spatial*
	Anachronisme	Disjonction
	Séquence	Adjacence
	Concomitance	Intersection
	Coïncidence	Imbrication
	Parallélisme	Recouvrement

Tableau 10 : Compositions spatiale et temporelle des objets mixtes.

Ainsi faisant, nous explicitons dans notre modèle le degré d'indirection, c'est-à-dire la distance, dans l'espace ou le temps, entre l'outil et l'objet sur lequel il agit [Beaudouin-Lafon, 2004]. Nous rapprochons également cette notion de relation entre objets à la compatibilité perceptuelle de [Dubois, 2001]. Il s'agit également d'un aspect (au niveau extrinsèque) de l'incarnation de [Fishkin, 2004] et de la composition spatiale entre éléments de [Fitzmaurice et al., 1995].

2.2.5. Caractérisation des métaphores d'interaction

En littérature, une métaphore est une figure de rhétorique qui consiste à donner à un mot un sens qu'on ne lui attribue qu'implicitement par un rapport de ressemblance établi par l'intelligence ou l'imagination. En informatique, nous trouvons des métaphores d'interaction, comme la métaphore de bureau des interfaces graphiques qui donne à l'interface l'aspect d'un bureau, avec des documents et des dossiers pour représenter la hiérarchie de fichiers, et des outils pour représenter les applications.

2.2.5.1. Métaphores de nom

Nous considérons l'aspect de la métaphore défini dans [Fishkin, 2004] : La métaphore de nom est définie par la phrase « un <X> dans le système est comme un <X> dans le monde réel ». Pour caractériser la métaphore de nom appliquée à un objet mixte dans le contexte de son application, nous considérons les relations entre les propriétés physiques de l'objet mixte et la tâche accomplie. Nous étudions la façon dont les propriétés physiques reflètent la tâche accomplie avec l'objet mixte. Par exemple, nous considérons le Phicon dans le système du Tangible Geospace de la Figure 63 qui a la forme du dôme du MIT : la forme représente donc un des paramètres de la tâche, puisqu'en déplaçant le phicon sur la table, l'utilisateur fait correspondre l'emplacement du dôme sur la carte avec l'emplacement de l'outil. La Figure 75 montre comment caractériser cette métaphore avec le modèle d'interaction mixte avec l'exemple du Phicon. Au contraire dans l'application NavRna de la Figure 66, il n'y a pas de métaphore de nom, puisque le jeton bleu a une forme générique qui n'a pas de rapport avec la tâche.

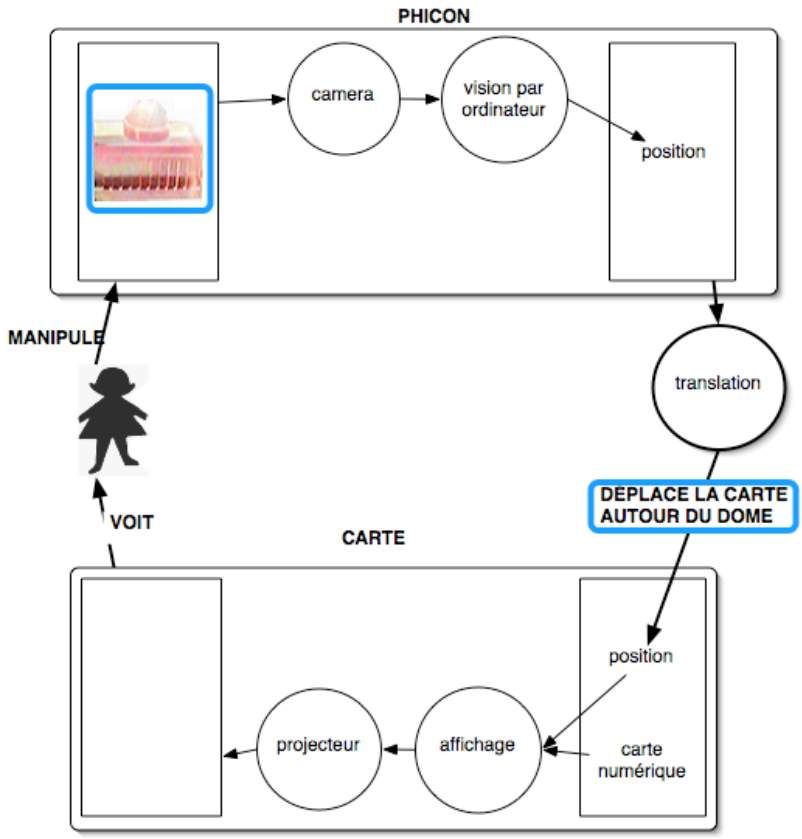


Figure 75 : Exemple de caractérisation de la métaphore de nom avec le modèle d'interaction mixte : le Tangible Geospace et sa métaphore se rapportant au paramètre de la tâche.

Au lieu de considérer seulement le paramètre de la tâche, la métaphore de nom peut aussi mettre en rapport les propriétés physiques de l'objet mixte avec la tâche elle-même. Comme nous pouvons le voir dans la Figure 57, les propriétés physiques de l'objet mixte, la forme de la gomme, représentent la tâche ou commande elle-même, l'effacement, au lieu du paramètre de la commande.

De plus, contrairement à [Fishkin, 2004], nous ne considérons pas seulement les métaphores avec le monde dit « réel ». En effet, la gomme ou le bouton de papier dans le DigitalDesk ne font pas référence au même monde ou domaine : tandis que la gomme fait référence au monde « réel », le bouton de papier fait référence au monde informatique, aux interfaces graphiques.
Le Tableau 11 récapitule la classification des systèmes exemples selon les métaphores de nom.

	Métaphore de nom		*Pas de métaphore de nom*
	Paramètre	**Tâche**	
Référence « physique »	Tangible Geospace, Dôme	DigitalDesk, gomme	PICO NavRNA Actuated workbench reacTable Music Bottle
Référence « numérique »	DigitalDesk, Bouton FILL		

Tableau 11 : Caractérisation des métaphores de nom dans les systèmes exemples.

2.2.5.2. Métaphores de verbe

Nous considérons l'aspect de la métaphore défini dans [Fishkin, 2004] : La métaphore de verbe est définie par la phrase « <X>-er l'objet dans le système est comme <X>-er dans le monde réel ». Pour caractériser la métaphore de verbe appliquée à un objet mixte dans le contexte de son application, nous étudions si les propriétés numériques acquises de l'objet mixte reflètent la tâche accomplie.

Par exemple dans la Figure 57, les mouvements reconnus de la gomme reflètent la tâche. Nous attirons l'attention du lecteur sur le fait que la gomme dans ce contexte d'interaction est caractérisée par une métaphore de verbe et de nom à la fois, mais nous pouvons les considérer indépendamment. Par exemple, la gomme pourrait être utilisée sans métaphore de verbe, en la posant sur le dessin pour gommer. Ou un morceau de papier pourrait être utilisé par frottements comme une gomme, et nous aurions ainsi une métaphore de verbe sans métaphore de nom. C'est l'exemple du bouchon des *music bottles*, qui est caractérisé par une métaphore de verbe sans métaphore de nom. La Figure 76 montre comment caractériser cette métaphore avec le modèle d'interaction mixte avec l'exemple d'une bouteille musicale.

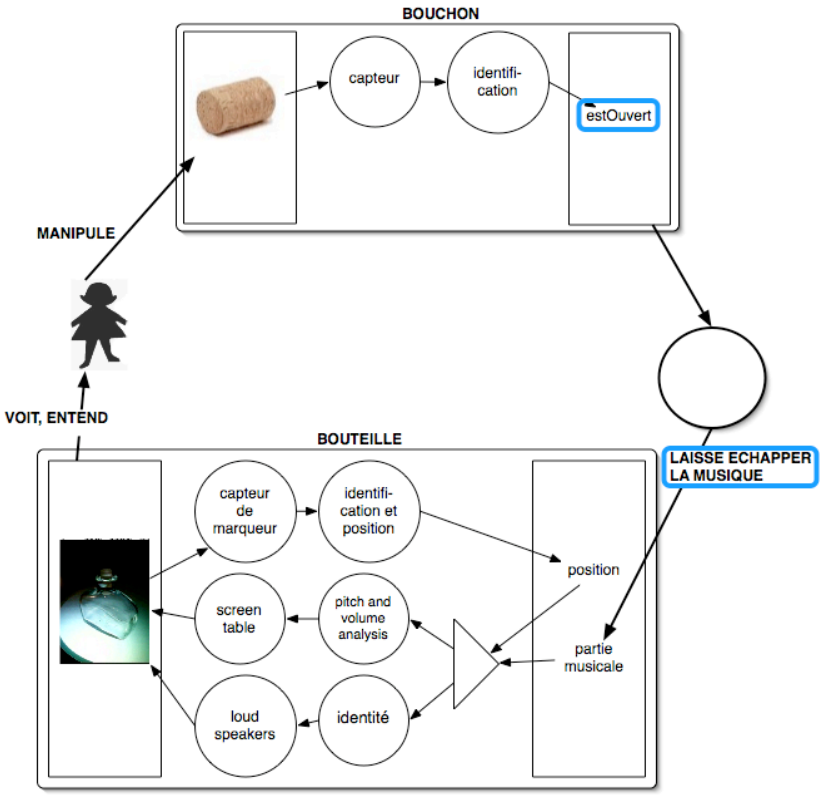


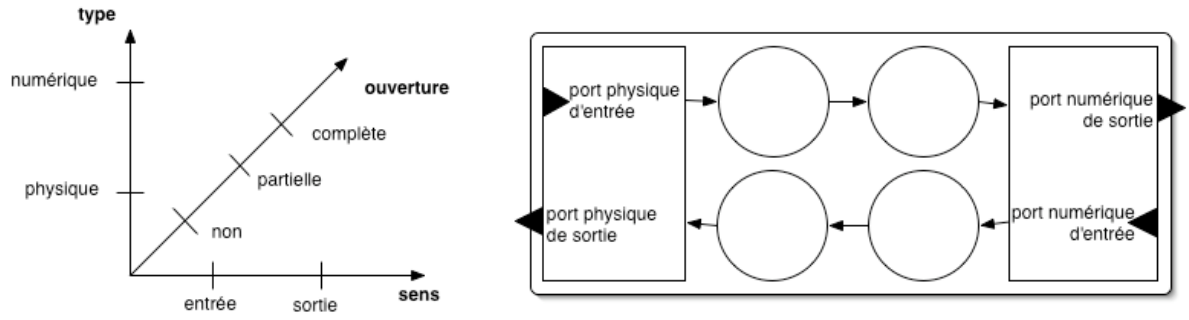
Figure 76 : Exemple de caractérisation de la métaphore de verbe avec le modèle d'interaction mixte: une *music bottle*.

	Métaphore de verbe	*Pas de métaphore de verbe*
Référence « physique »	DigitalDesk, gomme Music bottles	Tangible Geospace, Dôme PICO NavRNA Actuated workbench reacTable
Référence « numérique »	DigitalDesk, Bouton FILL	

Tableau 12 : Caractérisation des métaphores de verbe dans les systèmes exemples.

2.2.6. Caractérisation des ports d'un objet mixte

Enfin pour caractériser un objet mixte dans son contexte d'interaction (caractérisation extrinsèque) nous considérons ses ports. Ils permettent la communication avec d'autres éléments extérieurs à l'objet mixte. Un port est soit d'entrée ou de sortie du point de vue de l'objet. Comme nous mettons la partie physique et numérique d'un objet au même niveau, nous identifions des ports physiques et des ports numériques (Figure 77).



(**a**) (**b**)

Figure 77 : Typologie des ports d'un objet mixte.

2.2.6.1. Ports d'entrée physiques

Les ports d'entrée physiques d'un objet mixte sont ses propriétés physiques sur lesquelles l'utilisateur peut agir (Figure 77 en haut à gauche). Cette caractérisation des propriétés physiques est à rapprocher de l'affordance [Norman, 1999], qui sont les propriétés physiques d'un objet mixte sur lesquelles l'utilisateur peut agir.
Les ports d'entrée physiques sont une caractéristique extrinsèque, car ils nécessitent d'être reconsidérés dès que le contexte d'utilisation est modifié. En effet, si dans NavRNA un utilisateur pose de quoi écrire sur la table, alors les jetons ne peuvent aller sur cette partie de la table.

Cette caractéristique nous permet de modéliser les contraintes comme identifiées dans [Shaer et al., 2004]. En effet dans le contexte de l'application, certains états possibles des propriétés physiques peuvent être bloqués par des contraintes externes à l'objet mixte. Par exemple, la position physique des objets de PICO est modifiable par l'utilisateur, mais certaines positions sont impossibles à atteindre, à cause de la table, ou d'un objet lourd posé sur l'objet pour l'empêcher de bouger. Un port d'entrée physique peut ne pas être complètement ouvert :

1. À cause de limitations technologiques [Patten, Ishii, 2007], comme le cas de la table qui délimite les déplacements des cubes de la reacTable dans le champ de la caméra.
2. Par volonté de l'utilisateur [Patten, Ishii, 2007], comme dans PICO quand un utilisateur pose un objet lourd dessus pour l'empêcher de bouger.

2.2.6.2. Ports d'entrée numériques

Les ports d'entrée numériques d'un objet mixte sont ses propriétés numériques sur lesquelles le système peut agir, via un langage d'interaction. Par exemple, les jetons bleus de NavRNA et le Phicon du Tangible Geospace n'ont pas de port d'entrée numérique, au contraire des objets de la reacTable : La Figure 78 montre un exemple avec un oscillateur à basse fréquence (objet 2 à Figure 78) qui contrôle l'amplitude d'un filtre sonore passe-bas (objet 1 à Figure 78). Dans cet exemple, le système peut modifier les propriétés numériques de l'objet 1 grâce à l'objet 2 que l'utilisateur a ajouté sur la table près de l'objet 1 pour contrôler son port d'entrée numérique : l'amplitude du signal.

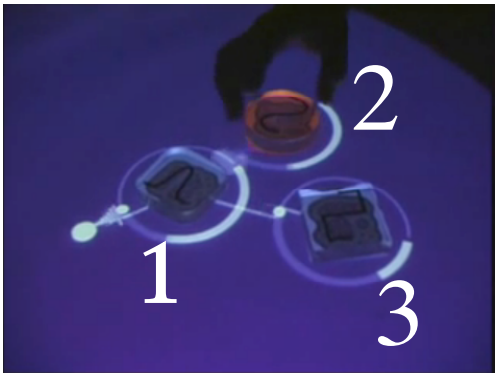


Figure 78 : ReacTable : Un oscillateur à basse fréquence (objet 2, circulaire et rouge avec une sinusoïde représentée dessus) qui contrôle un filtre sonore passe-bas (objet 1, bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus).

Nous introduisons avec cette caractéristique une nuance dans la configurabilité des outils [Underkoffler, Ishii, 1999] [Holmquist et al., 1999] [Fitzmaurice et al., 1995] [Shaer, 2004] [Renevier, 2004]. Ici il ne s'agit pas de remplacer une partie de l'outil, comme sa partie numérique ou ses modalités de liaison, mais d'ajuster un paramètre de cet outil. Ajuster une propriété de l'outil peut être fait « directement » par l'utilisateur via un port d'entrée physique (comme avec l'*actuated workbench* avec la caméra et la vision par ordinateur) ou indirectement via un outil et un port d'entrée numérique (comme avec l'*actuated workbench* avec la trackball).
De façon symétrique au côté physique, certaines valeurs pour les propriétés numériques peuvent être bloquées par l'extérieur de l'objet mixte. En effet, les langages d'interaction peuvent contraindre les propriétés numériques. Nous identifions donc la notion de contrainte numérique. Par exemple, dans Carcade que nous avons présenté au Chapitre 2 page 25, l'utilisateur, en voiture avec un ordinateur sur les genoux et une caméra placée face à la fenêtre de la voiture, manipule un avion dans un jeu où les éléments du paysage qui défile forment le décor du jeu. Dans ce jeu, le décor est mixte, et constitue une contrainte numérique pour l'avion.

2.2.6.3. Ports de sortie physiques

Les ports de sortie physiques d'un objet mixte sont ses propriétés physiques que l'utilisateur peut percevoir. Par exemple, si nous considérons les propriétés physiques générées correspondant à la projection sur la table dans les systèmes exemples, ces propriétés physiques peuvent être des ports de sortie qui peuvent être partiellement ouverts, selon le type de projection (venant de devant ou de l'arrière). En effet, avec une projection par le dessus, comme avec le DigitalDesk, l'*actuated workbench* et PICO, la main de l'utilisateur peut masquer une partie des propriétés physiques. Au contraire, avec une projection venant du dessous de la table, comme avec la reacTable ou les *music bottles*, ces propriétés physiques sont des ports de sortie toujours ouverts.
Cette notion est à rapprocher du concept de nimbus : le nimbus est l'espace où est dirigée la présence d'une entité, par exemple l'espace d'où l'entité peut être vue (Chapitre 3 page 49). Nous lui ajoutons une nuance due à son caractère extrinsèque : un nimbus dans le contexte de l'interaction.

2.2.6.4. Ports de sortie numériques

Les ports de sortie numériques d'un objet mixte sont ses propriétés numériques exploitables par un langage d'interaction en entrée. Par exemple, la position numérique est un port de sortie pour la gomme et le bouton *FILL* du DigitalDesk, les jetons de NavRNA, le Phicon du Tangible Geospace, les objets de la reacTable, car leur position numérique est utilisée par le langage d'interaction pour interagir avec l'objet de la tâche. Au contraire pour les *music bottles*, la position numérique est utilisée seulement à l'intérieur de l'objet pour la projection de couleur. Elle n'est pas utilisée pour une réponse vers l'outil (le bouchon).
Le port de sortie numérique explicite la notion de mouvements souhaités [Benford et al., 2005], comme défini dans la section 1.1.7 du Chapitre 3 (page 50). Nous rappelons ici que les mouvements

attendus constituent dans notre modèle deux caractéristiques : (1) intrinsèque à l'objet et liée à l'affordance et (2) extrinsèque à l'objet liée au port d'entrée physique. Les mouvements captures, quant à eux, constitue une propriété intrinsèque de l'objet mixte liée aux propriétés physiques capturées, tandis que les mouvements souhaités constituent une propriété extrinsèque : le port de sortie numérique.

La Figure 77 (a) montre la typologie des ports selon trois axes : un port peut être de type physique ou numérique. Il peut être complètement ouvert, partiellement ouvert ou fermé. Enfin nous trouvons des ports d'entrée et des ports de sortie. La Figure 77 (b) montre ces ports mis en évidence dans un objets mixte : en haut à gauche un port d'entrée physique, en bas à gauche un port de sortie physique, en haut à droite un porte de sortie numérique, et en bas à droite un port d'entrée numérique.

2.3. Synthèse des caractéristiques

Nous résumons dans le Tableau 13 les caractéristiques de notre modèle selon les deux points de vues adoptés : caractéristiques intrinsèques et extrinsèques.

Intrinsèques	Extrinsèques
Affordance des propriétés physiques	Ports d'entrée physique
Couplage des modalités de liaison	Ports de sortie physique
Propriétés physiques à rebond	Métaphore(s) de Nom
Propriétés physiques captées et Générées	Ports d'entrée numériques
Caractérisation spatiale et temporelle des propriétés physiques	Ports de sortie numérique
	Métaphore de verbe
Propriétés numériques acquises et perceptibles	Rôle
Propriétés numériques à rebond	Langages d'interaction
Dispositifs de liaison	Caractérisation spatiale et temporelle des objets entre eux
Langages de liaison	
Degré d'intégration	
Multimodalité des liaisons	

Tableau 13 : Résumé des caractéristiques intrinsèques et extrinsèques.

Ces caractéristiques nous ont permis de classer des systèmes très similaires. La Figure 79 récapitule la classification des sept systèmes exemples grâce aux caractéristiques que nous venons de présenter.

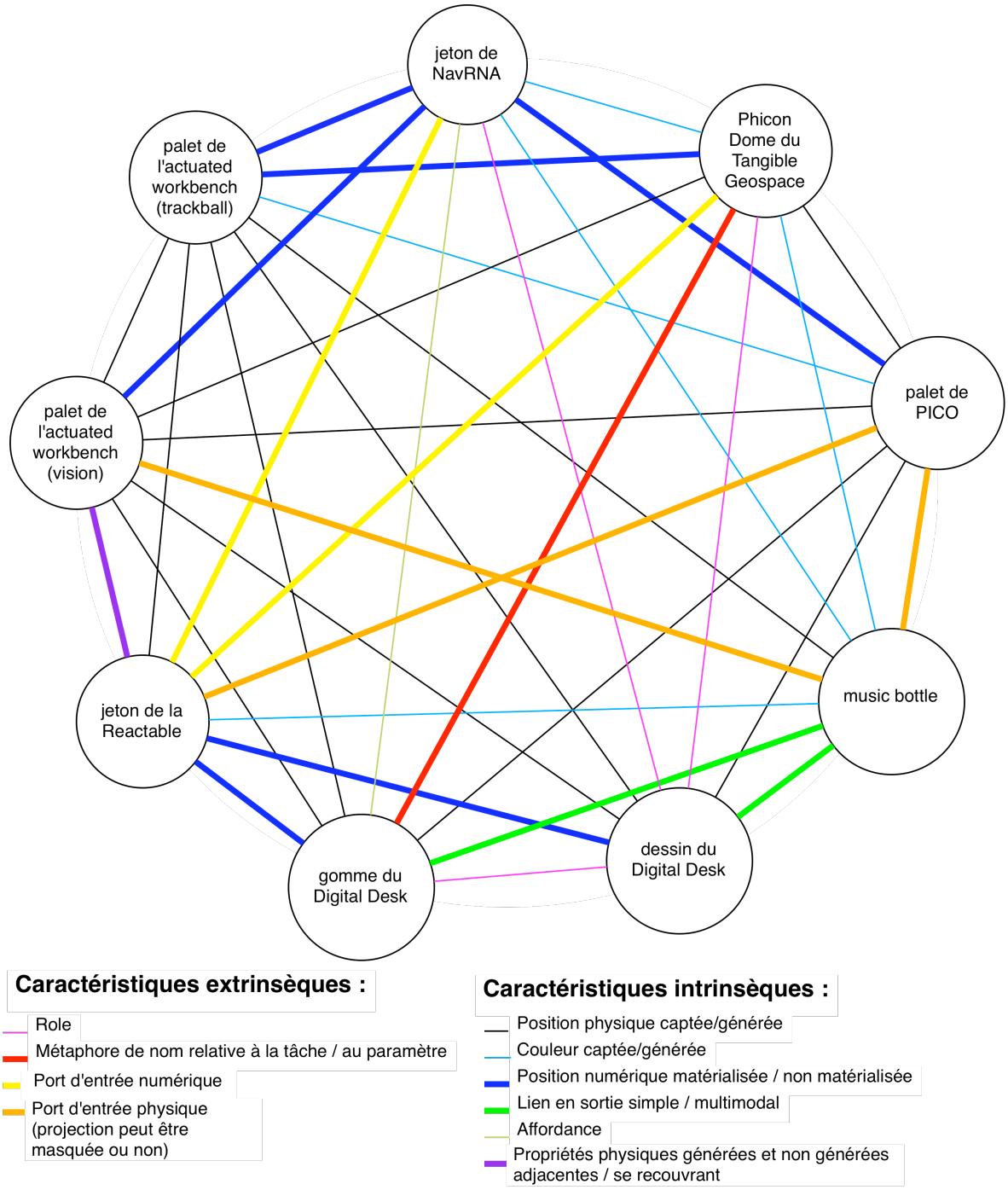


Figure 79 : Classification des exemples : Chaque système appartient à une unique catégorie. Dans ce diagramme, par souci de clarté, nous ne montrons qu'une différence basée sur une caractéristique entre chaque système, bien que plusieurs caractéristiques nous permettent de les distinguer.

Outre son pouvoir descriptif, nous avons donc montré le pouvoir comparatif de notre modèle en classant des interfaces mixtes que les modèles existants du chapitre précédent ne permettaient pas de différentier (section 2.3 du Chapitre 3).
Nous attirons l'attention du lecteur sur le fait que les carences évoquées dans le chapitre précédent sont comblées par le pouvoir taxinomique de notre modèle de l'interaction.

Afin de mesurer la contribution de ces caractéristiques, nous les analysons plus précisément selon deux axes. Tout d'abord nous nous inspirons de [Nigay, 1997, p.91]. Focalisant sur l'architecture

logicielle, quatre niveaux sont identifiés dans [Nigay, 1997, p.91] pour définir si une propriété ergonomique est vérifiée par un système ayant une certaine architecture logicielle :

1. Construire le système avec cette architecture implique que la propriété est vérifiée.
2. Il est possible de faire en sorte que la propriété soit vérifiée, en fournissant effort et/ou connaissance.
3. Il est possible d'évaluer le niveau de vérification de la propriété avec cette architecture.
4. On ne peut rien dire, la propriété ne s'exprime pas dans l'architecture.

Nous adaptons ces niveaux au problème qui nous concerne : nous caractérisons l'effort qu'il est nécessaire d'effectuer pour trouver la valeur d'une caractéristique dans le modèle d'interaction mixte. Nous identifions trois possibilités :

1. *Explicite* : On observe directement dans la modélisation la valeur de la caractéristique. Le modèle explicite la valeur de ce concept.
2. *Indirectement exprimée* : Il est possible de trouver la valeur de la caractéristique, en fournissant effort et/ou connaissance.
3. *Absente* : Il n'est pas possible de trouver la valeur de la caractéristique au sein du modèle (mais d'une autre façon).

Outre le degré d'explicitation de la caractéristique par notre modèle d'interaction mixte, nous étudions aussi si la caractéristique est nouvelle, étendue, ou reprise de l'existant. Ceci constitue notre deuxième axe d'étude des caractéristiques.

Notre espace d'évaluation des caractéristiques du modèle considère donc deux axes présentés à la Figure 80. Dans cet espace, les contributions les moins importantes (capitalisée et/ou absente) sont en bas et/ou à gauche en rouge, et les plus importantes (explicite et/ou nouvelle) sont placées en haut et/ou à droite en bleu.

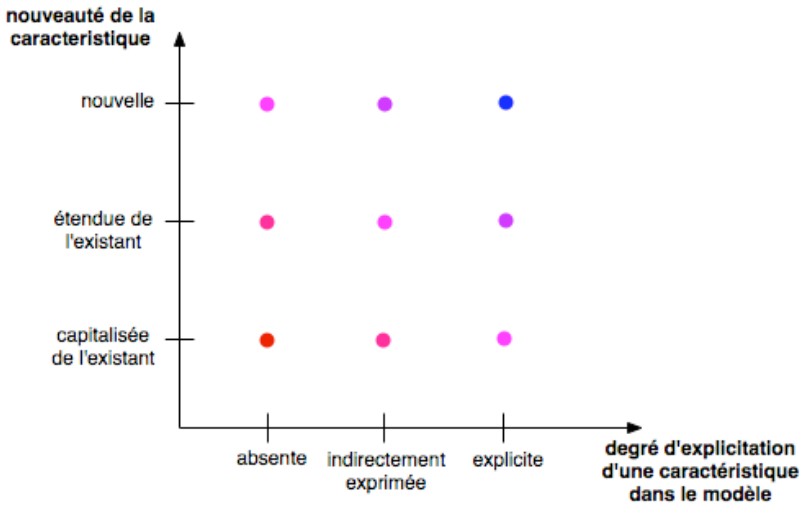


Figure 80 : Espace d'évaluation des caractéristiques du modèle.

Nous présentons notre analyse des caractéristiques présentées dans ce chapitre selon cet espace d'évaluation dans le Tableau 14.

Le Tableau 15 présente, selon ce même espace d'évaluation, les caractéristiques existantes présentée au Chapitre 3. Parmi les caractéristiques existantes qui n'ont pas été capitalisées, nous trouvons des caractéristiques que nous n'avons pas étudiées au regard du modèle, mais qui font néanmoins partie des perspectives à plus ou moins court terme de ce travail. De plus, certaines caractéristiques existantes sont présentes dans ce tableau bien que non citées dans ce chapitre : elles sont explicites dans le modèle d'interaction, et font plutôt partie de la partie description de l'interaction. C'est le cas par exemple de l'exécution ou évaluation augmentée [Dubois, 2001], qui est explicitée dans la modalisation.

	Absente	Indirectement exprimée	Explicite
Nouvelle		Propriétés numériques à rebond	Propriétés numériques acquises et perceptibles Ports d'entrée numérique
Étendue de l'existant		Degré d'intégration Caractérisation spatiale et temporelle des propriétés physiques Caractérisation spatiale et temporelle des objets entre eux	Propriétés physiques captées et générées Ports de sortie physique Langages d'interaction Métaphore(s) de nom
Capitalisée de l'existant		Couplage des modalités de liaison Affordance des propriétés physiques Propriétés physiques à rebond	Dispositifs de liaison Langages de liaison Liaisons multimodales Rôle Ports d'entrée physique Ports de sortie numérique Métaphore de verbe

Tableau 14 : Analyse des caractéristiques du modèle d'interaction mixte présentées selon deux axes.

	Absente	**Indirectement exprimée**	**Explicite**
Nouvelle			
Étendue	Propriétés ergonomiques [Bastien, Scapin, 1993] Réalité/Virtualité augmentée [Dubois, 2001] ou représentation de l'interaction [Fitzmaurice et al., 1995] Mode d'interaction [Renevier, 2004] Durée d'interaction élémentaire [Fitzmaurice et al., 1995] Granularité [Fitzmaurice et al., 1995] But d'introduction de l'informatique [Jacob et al., 2008]	Type de langage de l'adaptateur [Dubois, 2001] Représentation [Dubois, Gray, 2007] Axe entrée/sortie [Fitzmaurice et al., 1995] Composition spatiale [Fitzmaurice et al., 1995] Incarnation [Fishkin, 2004] Compatibilité et continuité perceptuelles [Dubois, 2001] Temporalité du lien [Fitzmaurice et al., 1995] *Degré d'intégration [Beaudouin-Lafon, 2000]* Propriété physique à rebond [Dix et al., 2007] Outils reconfigurables [Underkoffler, Ishii, 1999] Conteneur [Holmquist et al., 1999] Assignation [Fitzmaurice et al., 1995]	Métaphore de Nom [Fishkin, 2004]
Capitalisée		Nombre de dimensions de l'adaptateur [Dubois, 2001] Position [Mackay, 1996] Medium [Benford, Fahlen, 1993] Nimbus [Benford, Fahlen, 1993] Focus [Benford, Fahlen, 1993] Mouvements capturés et leurs caractéristiques [Benford et al., 2005] Mouvements attendus [Benford et al., 2005] *Caractérisation des langages [Bernsen]* *Caractérisation des langages [Vernier, Nigay, 2000]* *Composition des modalites [Vernier, Nigay, 2000]* *Affordance [Norman, 1999]* *Instrument et Objet du domaine [Beaudouin-Lafon, 2000]* Outil et Objet [Dubois, 2001] Eléments O et T [Chalon, 2004] Objets verbes [Underkoffler, Ishii, 1999] Outil et Token [Holmquist et al., 1999] Contraintes [Shaer et al, 2004][Patten, Ishii, 2007]	*Dispositif et langage d'interaction [Nigay, 1994]* Adaptateurs [Dubois, 2001] Indice entrée/sortie de l'adaptateur [Dubois, 2001] Mécanisme de capture [Dubois, Gray, 2007] Adaptateurs [Benford, Fahlen, 1993] Métaphore de verbe [Fishkin, 2004] Mouvements souhaités [Benford et al., 2005] Exécution et/ou évaluation augmentée [Dubois, 2001] Nombre utilisées en parallèle [Fitzmaurice et al., 1995]

Tableau 15: Analyse des caractéristiques issues de travaux existants au regard du modèle d'interaction mixte : les caractéristiques en italique viennent d'autres domaines que les interfaces mixtes.

Ce modèle d'interaction mixte présente donc encore des imperfections : certaines propriétés ne sont pas capitalisées ou d'autres pas explicites (Tableau 14). Ceci constitue des travaux à court terme, afin que les caractéristiques soient toutes prises en compte par notre modèle. Cependant, nous constatons grâce à ces deux tableaux que le modèle, grâce à sa structure globale solide, est un bon candidat pour la réunion et l'extension de ces espaces de conception existants.

3. Synthèse

Dans ce chapitre, nous avons présenté notre modèle d'interaction mixte et ses caractéristiques. Nous avons montré dans quelle mesure il permet d'aller plus loin que les travaux existants.

Nous avons souligné dans la première partie que le modèle permet de décrire un grand nombre de systèmes existants. Nous l'avons montré en considérant des exemples balayant un large éventail de possibilités. Il convient néanmoins de noter que certaines interfaces mettent l'accent sur un élément particulier du modèle (comme les propriétés physiques dans [Murray-Smith et al., 2008]) et nécessiteraient alors plus de détails pour une meilleure description.
Dans une seconde partie, nous avons éprouvé le pouvoir comparatif du modèle grâce à des exemples plus difficiles à distinguer. Au-delà de l'aspect taxinomique, nous avons évalué le pouvoir comparatif du modèle en le situant par rapport aux travaux existants. Nous avons souligné la capitalisation d'un grand nombre de caractéristiques pertinentes et validées par la communauté scientifique. De plus, nous avons affiné et étendu certaines caractéristiques, et nous en avons introduites de nouvelles.

Mais ces pouvoirs descriptifs et taxinomiques ne sont qu'une facette d'un modèle d'interaction. En effet un tel outil vise principalement à soutenir la conception en définissant un cadre à l'exploration de l'espace de conception. Nous présentons dans le chapitre suivant comment nous avons évalué ce pouvoir génératif du modèle d'interaction mixte.

Chapitre 5 : Carnet de route et évaluation du modèle d'interaction mixte

Au-delà de son pouvoir unificateur, il convient de démontrer l'intérêt du modèle du point de vue de son utilisation. Nous rappelons que dans [Beaudoin-Lafon, 2004], la qualité d'un modèle est reliée à trois pouvoirs :

1. Le pouvoir descriptif a été validé par la modélisation des systèmes existants dans la première partie du chapitre précédent.
2. Le pouvoir évaluatif ou taxinomique a été validé par la classification des systèmes existants dans la seconde partie du chapitre précédent.
3. Le pouvoir génératif est la raison d'être d'un modèle d'interaction [Beaudoin-Lafon, 2004]. Nous avons donc apporté une attention particulière à son évaluation. Ce chapitre présente le carnet de route de l'élaboration du modèle d'interaction mixte et son évaluation.

Nous abordons tout d'abord les problèmes liés à la validation d'un tel outil conceptuel, et nous présentons ensuite les études que nous avons menées, autant conceptuelles qu'expérimentales.

1. Validation d'un modèle d'interaction

Afin de mettre à jour les difficultés liées à l'évaluation, nous présentons d'abord les validations faites pour les modèles existants présentés au Chapitre 3, ainsi que les cadres de validation proposés dans la littérature. Nous les analysons ensuite pour en dégager les apports et les limites, afin d'enrichir notre propre processus d'évaluation.

1.1. Approches existantes

Notons que très peu d'études présentées au Chapitre 3 ont focalisé sur l'évaluation de leur contribution. Néanmoins la littérature révèle différentes formes d'évaluation.

Nous trouvons tout d'abord des démonstrations conceptuelles. Par exemple dans [Fishkin, 2004], le pouvoir comparatif de la taxonomie présentée est validé en montrant qu'elle inclut les autres et révèle des zones de l'espace de conception non explorées. Dans [Shaer et al., 2004], les auteurs proposent une validation pratique du pouvoir descriptif de leur travail en l'appliquant sur un échantillon « représentatif » d'interfaces existantes. Au-delà de la validation ponctuelle d'un outil conceptuel, nous trouvons dans la littérature des cadres d'évaluation conceptuels. Par exemple, le cadre d'évaluation des dimensions cognitives des notations [Blackwell, Green, 2000] se présentent sous forme d'un questionnaire d'inspection. D'autres approches, comme [Olsen, 2007], peuvent être appliquées aux outils conceptuels, même si elles émanent de l'évaluation des outils logiciels, car leur raisonnement est pertinent pour les outils de conception.

Toutes ces propositions de validation (conceptuelles) ont en commun qu'elles ne reposent pas sur l'expérience avec des utilisateurs de l'outil de conception. Au contraire, d'autres travaux proposent de telles évaluations expérimentales. Nous trouvons aussi dans [Dubois et al., 2006] une proposition d'articulation entre l'utilisation d'une approche conceptuelle et d'un groupe focalisé pour la conception. Toutefois, les auteurs n'en relatent pas l'expérience.

Dans [Anranda et al, 2007] un cadre d'évaluation des notations qui doit être appliqué par l'expérience est présenté. Ce cadre vise l'évaluation de la compréhensibilité d'une notation. Les auteurs proposent une méthode de construction du protocole d'évaluation et identifient les dimensions sur lesquelles porte l'évaluation :

- Les dimensions influant sur la compréhensibilité, comme le type de tâche, l'expertise du modèle qu'ont les utilisateurs, la taille du problème.
- Les dimensions sur lesquelles influent la compréhensibilité, comme l'exactitude/justesse de ce qui a été compris, le temps requis perçu (subjectif) pour comprendre la représentation, la confiance (subjective) que les utilisateurs ont dans ce qu'ils ont compris, la difficulté perçue (subjective) que les utilisateurs ont eue pour accéder à l'information contenue dans les schéma.

Nous trouvons enfin une expérience de validation dans [Iachello, 2006 : p.109(125)] : l'auteur a construit une méthode de conception des systèmes ubiquitaires tenant compte des questions de sécurité et de respect de la vie privée et des données personnelles. Il a réalisé une évaluation pilote avec six étudiants volontaires. Elle vise à évaluer la compréhensibilité, l'utilisabilité, et la stabilité

des résultats entre les concepteurs. L'évaluation finale a eu lieu avec les 48 étudiants d'un cours d'introduction à l'Interaction Homme-Machine. Dans son évaluation, il compare expérimentalement sa méthode avec deux autres. Elle vise à évaluer la qualité des applications conçues (au regard des questions de sécurité) et les comparer. Pour cela, l'auteur a évalué les travaux des étudiants avec 25 métriques, comme par exemple la note obtenue par les étudiants ou le nombre de questions de conception considérées.
Toutes ces approches ont leurs intérêts et leurs limites. Nous en présentons maintenant notre analyse.

1.2. Apports et limites

Les limites de ces approches résident dans leur impuissance à valider rigoureusement une contribution. En effet, les méthodes traditionnelles d'évaluation expérimentale se heurtent ici à la complexité du problème :

- La tâche est complexe (concevoir) et parfois longue,
- Ce que l'on veut mesurer est large et souvent mal défini (utilité ? qualité du système final conçu ? facilité d'utilisation ? etc.),
- Les utilisateurs ont des profils hétérogènes (informaticien, ergonome, designer, artiste, etc.).

De la même façon que pour prouver le pouvoir descriptif ou comparatif du modèle, il nous sera difficile d'en évaluer le pouvoir génératif. En effet, comment prouver rigoureusement qu'un modèle d'interaction permet effectivement de décrire toutes les interfaces qu'il prétend pouvoir décrire ? Il n'est pas envisageable de les décrire de façon exhaustive. Et comment trouver un échantillon *représentatif* de ces interfaces ? Les approches proposées sont empiriques, et, même si elles permettent de convaincre, elles ne prouvent rien. De même, comment prouver rigoureusement qu'un modèle d'interaction permet effectivement de comparer toutes les interfaces qu'il prétend pouvoir comparer ? Pour évaluer le pouvoir génératif, des difficultés sont encore là. Comment apprécier le pouvoir génératif d'un modèle d'interaction, et le comparer avec ceux des autres modèles afin de prouver sa contribution ? Les variables à contrôler sont trop nombreuses. Comment être sûr que les résultats ne sont pas biaisés par l'expertise des concepteurs ? Comment prouver qu'une solution est conçue avec ou sans l'aide de l'outil ?

De plus, la difficulté de l'évaluation réside également dans les moyens logistiques nécessaires à la mise en place d'une expérience. Les difficultés classiques rencontrées pour l'évaluation des techniques d'interaction se trouvent ici exacerbées : trouver les sujets au profil adapté, trouver le temps et la durée nécessaire pour l'évaluation, etc.

À ces difficultés s'ajoute qu'il n'est pas encore consensuel qu'une telle démonstration soit nécessaire. En effet il n'est pas évident qu'une évaluation suffise à convaincre les concepteurs d'adopter effectivement un outil.

Pourtant, avant même d'effectuer une évaluation du modèle d'interaction mixte, il convient de se demander quel en est l'objectif. Notre objectif est l'amélioration de l'utilité et de l'utilisabilité de l'outil. Une utilisation concrète et non contrôlée du modèle et des évaluations, même imparfaites et critiquables, peuvent néanmoins nous apporter des indices quant à ces questions. Les approches de validation présentées ne permettent pas une validation rigoureuse d'un cadre de conception, mais leur intérêt réside dans l'aperçu qu'elles donnent des capacités et limites d'un outil conceptuel. Tout en restant conscients des limites de ces approches, elles peuvent donc servir de support à l'évolution du modèle évalué.

À partir des premières versions du modèle d'interaction mixte, et jusqu'à sa dernière version, nous avons utilisé concrètement son pouvoir génératif. Nous rappelons ici que le pouvoir génératif n'est pas (nécessairement) la capacité du modèle à aider la conception de *nouvelles* formes d'interaction, mais la capacité (nécessaire) d'explorer l'espace de conception de façon systématique. Nous

présentons d'abord les évaluations conceptuelles effectuées, puis le carnet de route des utilisations du modèle et les études expérimentales effectuées pour l'évaluer.

2. Évaluations conceptuelles

Par l'utilisation de l'adjectif conceptuelle, nous entendons que ces formes d'évaluation n'ont pas été effectuées par l'expérience. Au contraire, nous avons vérifié « sur le papier » si le modèle d'interaction mixte respectait les conseils trouvés dans la littérature. Les hypothèses faites sur le modèle d'interaction mixte que nous cherchions à valider ainsi sont :
H_1 : « Le Modèle d'Interaction Mixte aide l'exploration de l'espace de conception. »
H_2 : « Le Modèle d'Interaction Mixte facilite l'utilisation conjointe des approches existantes. »
Avant d'aborder l'évaluation de H_2, nous présentons maintenant comment nous avons évalué H_1 avec les dimensions cognitives des notations de [Blackwell, Green, 2000].

2.1. Inspection selon les dimensions cognitives des notations

Pour vérifier conceptuellement H_2, nous utilisons la méthode d'inspection décrite dans [Green][Blackwell, Green, 2000], appelée « les dimensions cognitives des notations ». Cette méthode préconise que pour la conception exploratoire, les concepteurs ont besoin de notations de faible **viscosité**, faibles **contraintes**[26], forte **visibilité**, forte **lisibilité du rôle**[27]. Nous détaillons la définition de ces caractéristiques ainsi que notre analyse du modèle d'interaction mixte au regard de ces caractéristiques.

La **viscosité** est la résistance au changement. Nous avons construit le modèle d'interaction mixte de telle façon que, quand un concepteur veut faire un changement dans la conception, il n'a qu' à adapter la partie qui l'intéresse et éventuellement ses parties voisines.
Par exemple, nous verrons dans ce chapitre l'exemple de la conception d'ORBIS. Pour ce système, si le concepteur décide de matérialiser la propriété numérique *top*, il n'a qu'à adapter la modalité de liaison en sortie. Nous avons donc conçu le modèle d'interaction mixte pour que la viscosité soit faible.

Les **contraintes**[28] sont les contraintes dans l'ordre de conception. Nous avons construit le modèle d'interaction mixte de telle façon que le concepteur puisse choisir l'ordre dans lequel il conçoit : de la même façon qu'une architecture logicielle modulaire autorise la modificabilité du logiciel, l'identification des niveaux d'abstraction dans le modèle d'interaction mixte permet au concepteur de cibler l'élément qu'il veut concevoir.
Par exemple, il peut commencer à haut niveau d'abstraction (extrinsèque) en décidant quels seront l'outil et l'objet de la tâche, puis concevoir les parties à plus bas niveau d'abstraction (intrinsèque) en décidant quelles seront les propriétés physiques et numériques, ainsi que les modalités de liaison. Il peut aussi faire l'inverse et commencer par concevoir un objet, puis l'interaction entre les objets. Dans l'exemple d'ORBIS qui sera présenté dans ce chapitre, nous avons choisi d'abord de concevoir à haut niveau d'abstraction puis à bas niveau d'abstraction. Au contraire dans l'exemple de Snap2Play, nous avons d'abord conçu complètement l'objet de la tâche avant de passer à une conception à plus haut niveau d'abstraction (niveau interaction), pour enfin concevoir les outils et leurs composantes.

La **visibilité** est la possibilité de voir ou trouver les éléments de la notation facilement, quand ils sont conçus ou modifiés. Nous avons construit le modèle d'interaction mixte de telle façon qu'on

[26] *premature commitment*
[27] *role expressiveness*
[28] *premature commitment*

puisse voir tous les éléments à tout instant. Les niveaux intrinsèque et extrinsèque sont imbriqués et visibles simultanément. Trouver une partie du design extrinsèque doit être facile avec l'organisation en outil et objet de la tâche, interagissant avec un langage d'interaction. Ces éléments sont mis en évidence en étant entourés comme un tout. Pourtant, certaines caractéristiques ne peuvent pas être vues directement sur le schéma, et demandent un minimum d'effort de la part du concepteur. Par exemple, la continuité ou discontinuité spatiale d'un objet est implicite dans un schéma. Un concepteur doit comparer explicitement des propriétés physiques afin de connaître leurs relations spatiales et savoir si elles sont, par exemple co-localisées ou adjacentes. Pourtant, si un concepteur doit comparer des éléments d'un schéma pour trouver la valeur d'une caractéristique, il peut le faire assez facilement puisque tous les éléments de bases sont visibles en même temps. La distribution entre caractéristiques explicites ou indirectement exprimée par le modèle a été étudiée au Chapitre 4 de façon exhaustive pour toutes les caractéristiques du modèle.

La **lisibilité du rôle**[29] est la facilitée avec laquelle la fonction d'un composant est déduite par les concepteurs. Afin de respecter cette qualité sans contraindre trop fortement, nous proposons de (1) regrouper les éléments bas niveau (intrinsèque) dans des conteneurs (objets mixtes), et nous proposons des formes pour les modalités de liaisons (cercle) et les propriétés (rectangles). Néanmoins, cette intention nécessite, comme les précédentes, d'être évalué expérimentalement avec des concepteurs, et cette inspection conceptuelle ne saurait suffire à leur validation. Cette inspection est donc complétée par des expérimentations décrites dans la section 3.

2.2. Inspection selon [Olsen, 2007]

Nous souhaitons aussi vérifier que le modèle d'interaction mixte facilite l'utilisation conjointe des approches existantes (H_2, pouvoir unificateur du modèle). Pour cela, nous utilisons la méthode décrite dans [Olsen, 2007]. Même si la méthode d'évaluation qu'il propose concerne les outils logiciels, nous l'appliquons aux modèles d'interaction. En effet, le raisonnement proposé est pertinent pour les outils de conception.
Nous évaluons le modèle d'interaction mixte pour un groupe de concepteurs en séance de travail, pour la tâche de conception exploratoire d'une interface mixte. Nous justifions de l'importance du problème par notre expérience : les systèmes étaient conçus de façon ad hoc, peu de traces des éléments de choix de conception étaient conservées. Nous étions souvent obligés de concevoir un nouveau système à partir de rien, et nous rencontrions souvent les mêmes problèmes à nouveau. Nous n'étions pas capables d'explorer l'espace de conception de façon systématique. Nous trouvions souvent une meilleure solution une fois le développement terminé. De plus, nous rencontrions souvent des problèmes de compréhension quand nous communiquions sur la conception elle-même. En tant que concepteurs de systèmes de réalité mixte, nous étions concernés de près par l'importance de ce problème.
C'est pour répondre à ces problèmes que nous avons construit le modèle d'interaction mixte. Pour évaluer le pouvoir unificateur du modèle, nous avons montré que :

1. Le modèle intègre au sein d'un canevas les travaux existants, comme nous l'avons présenté au Chapitre 4, section 2.3. Ces travaux en Interaction Homme-Machine ont été reconnus comme intéressants par leurs pairs.
2. Le modèle est pertinent pour la conception d'interfaces mixtes, puisque les caractéristiques identifiées permettent de faire la différence entre des systèmes existants (Chapitre 4 section 2.3).
3. Le modèle est également non trivial, car il affine, détaille, et généralise les caractérisations existantes, comme nous l'avons souligné en détail au Chapitre 4 à la section 2.3.

Selon [Olsen, 2007], ces trois points prouvent que le modèle d'interaction mixte simplifie les interconnexions entre les approches existantes.

Ces approches conceptuelles permettent de d'éviter tôt certaines erreurs. Par exemple, [Olsen, 2007] nous ont permis de vérifier quelles caractéristiques existantes sont intégrées dans le modèle.

[29] *role expressiveness*

Pourtant elles doivent nécessairement se compléter par des expériences de validation avec des concepteurs, afin de mieux cerner l'intérêt et les limites de notre approche. Après avoir présenté ces premières évaluations conceptuelles, nous présentons maintenant les expériences effectuées avec le modèle d'interaction mixte.

3. Carnet de route et études expérimentales

Pour l'évaluation expérimentale du pouvoir génératif du modèle, nous avons adopté une démarche itérative. Le modèle a été utilisé concrètement pour la conception de sept interfaces mixtes, qui ont permis de faire l'expérience du modèle dans des situations réelle de conception. Nous avons fait varier plusieurs variables au cours de ces différentes expériences, comme l'expertise des concepteurs, le domaine d'application ou la complexité de l'application. Ces études expérimentales se sont décomposées en cinq temps :

1. Nous avons tout d'abord éprouvé le modèle par des experts pour concevoir des interfaces mixtes :
    a. Le premier système conçu fut RAZZLE, avec des utilisateurs connaisseurs du modèle, et à partir d'un jeu mixte mobile et multimodal existant à améliorer.
    b. Le second système conçu fut Playground, une installation artistique, à concevoir entièrement, avec des utilisateurs connaisseurs du modèle.
    c. Le troisième système conçu fut Snap2Play, un jeu mixte et mobile sur téléphone portable à concevoir entièrement, avec des utilisateurs connaisseurs du modèle.
2. Dans un second temps, nous avons fait utiliser le modèle par des novices accompagnés d'expert(s) :
    a. Le premier système conçu ainsi fut ORBIS, un système pour le visionnage de photos à concevoir entièrement par des utilisateurs connaisseurs et non connaisseurs du modèle.
    b. Le second système fut Roam, un système mobile d'enregistrement sonore conçu entièrement par des utilisateurs connaisseurs et non connaisseurs du modèle.
3. Nous avons ensuite fait utiliser le modèle par un novice seul : Le système Yubi a été conçu par un utilisateur non expert du modèle.
4. Enfin nous avons effectué une évaluation avec protocole, sous la forme d'un groupe focalisé[30] instrumenté par le modèle d'interaction mixte. Dans le groupe, les concepteurs présentaient chacun des expertises différentes et ont conçu un système à partir d'une proposition initiale.
5. Enfin, grâce aux retours obtenus lors de ces évaluations sur des cas concrets de conception, nous avons mené une évaluation exploratoire de la compréhensibilité du modèle sous la forme de questionnaires.

Avant de présenter l'évaluation de la compréhensibilité du modèle (étape 5), nous présentons les expériences des étapes 1 à 4 et leurs résultats au regard de l'évaluation du pouvoir génératif du modèle d'interaction mixte.

3.1. Expériences de conception réelles

Nous présentons ici les expériences de conception concrètes, sans protocole, de

- RAZZLE, un jeu mixte mobile et multimodal existant à étendre,
- Snap2Play, un jeu mixte et mobile sur téléphone portable,
- Playground, une installation artistique,
- ORBIS, un système pour le visionnage de photos,
- Roam, un système mobile d'enregistrement sonore,
- Yubi, un système pour augmenter les objets du quotidien.

[30] focus group

Les expériences de conception de ces systèmes ont toutes renseigné l'évaluation du pouvoir génératif du modèle d'interaction mixte.

3.1.1. Conception par un expert seul

3.1.1.1. RAZZLE : un jeu de réalité augmentée multimodal et mobile

À la suite de MAGIC [Renevier et al., 2001] [Renevier, 2004] et de TROC [Nigay et al., 2003] [Renevier, 2004], deux systèmes de réalité mixte, collaboratifs et mobiles, le jeu RAZZLE a été développé et exposé comme démonstrateur de ce type de système lors de la fête de la science les 15 et 16 octobre 2004 (Figure 81). Le développement d'une première version a donc précédé le modèle d'interaction mixte. Par conséquent nous avons travaillé sur RAZZLE à partir d'une version déjà existante. Nous présentons maintenant cette version existante afin de mieux situer notre expérience.



Figure 81 : RAZZLE, version existante. Un enfant joue à la fête de la science et ramasse une pièce avec la modalité gestuelle.

3.1.1.1.1. La version existante de RAZZLE

La première version de RAZZLE est un jeu collaboratif pour deux joueurs. Le but du jeu est de ramasser les pièces d'un puzzle numérique qui sont ancrés dans le monde physique. Les joueurs doivent donc se déplacer pour explorer le terrain de jeu. RAZZLE appartient à la classe des systèmes de virtualité augmentée : des pièces d'origine numérique sont dispersées partout sur le terrain de jeu. L'utilisateur peut interagir avec les pièces mixtes grâce à la technique du « terrain augmenté » [Renevier, 2004]. Cette technique permet aux utilisateurs de voir des objets mixtes avec une localisation dans l'espace physique, et décider s'ils sont bien orientés et suffisamment proches de l'objet. Un joueur ne peut interagir qu'avec les quatre pièces qu'il cherche et il est le seul joueur à chercher ces pièces. Ces quatre pièces sont identifiées par la couleur (pièces aux bords verts à la Figure 82, par exemple). Une fois ramassées, les pièces sont « portées » par le joueur, jusqu'à ce qu'il les dépose dans une zone particulière du monde physique, commune aux deux joueurs. Cette zone physique s'appelle la « zone de dépose ». Les joueurs s'y déchargent automatiquement de leurs pièces de puzzle en entrant tout simplement dans la zone de dépose. Une fois déchargées, les pièces sont ajoutées au puzzle. Le jeu s'arrête quand le puzzle est complet.

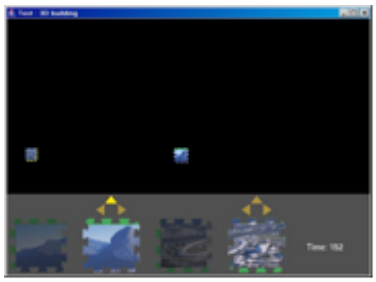


Figure 82 : Vue de ce que l'utilisateur voie dans les lunettes semi-tansparentes de RAZZLE, première version (les pixels noirs sont transparents).

Nous décrivons les plates-formes matérielle et logicielle de RAZZLE, pour ensuite détailler les modalités disponibles pour interagir avec les pièces de puzzle mixtes.
Chaque joueur dans RAZZLE porte un ordinateur portable où sont installés Java et la bibliothèque jogl (Open GL pour Java, https://jogl.dev.java.net). Le terrain de jeu est couvert par un réseau sans fil, ce qui permet à l'ordinateur portable du joueur d'être connecté au serveur central de RAZZLE. Des lunettes semi-transparentes sont également connectées à l'ordinateur portable. De plus, l'ordinateur portable possède un port série RS-232 pour connecter un capteur d'orientation XSENS MT9. La Figure 83 est le plan d'un joueur équipé du matériel décrit. À l'ordinateur portable est connecté la trackball blanche qu'il tient dans la main. La trackball lui permet de bouger un curseur affiché dans les lunettes par-dessus le monde physique et de cliquer. Ainsi l'utilisateur peut ramasser les pièces de puzzle. Enfin, nous utilisons la technique du magicien d'Oz pour simuler l'information sur la position du joueur. Dans la Figure 83, un compère suit le joueur sur le terrain de jeu pour mettre à jour sa position.



Figure 83 : RAZZLE, première version. L'utilisateur porte le matériel nécessaire, et un compère joue le rôle du Magicien d'Oz (au premier plan, à gauche).

La technique du « terrain augmenté » [Renevier, 2004] est basée sur la semi transparence des lunettes. À travers les lunettes, plus les pixels sont sombres, plus ils sont transparents (Figure 82). Le programme affiche donc dans les lunettes les éléments de la scène du jeu en 3D, utilisant jogl, auxquels s'ajoutent les objets du monde physique (portes, murs, etc.) qui sont déjà visibles à travers les lunettes. La Figure 82 illustre ce que le système affiche dans les lunettes semi transparentes. Puisque les pixels noirs sont transparents, l'utilisateur voit cette scène par-dessus le monde physique à travers les lunettes semi-transparentes. Dans l'interface de la Figure 82, l'affichage se divise en deux parties horizontales. La partie supérieure de l'interface correspond à la superposition des informations numériques au monde physique. On y voit deux pièces de puzzle selon le point de vue de l'utilisateur. La partie basse de l'interface correspond à un tableau de bord. Il indique au joueur les quatre pièces qu'il doit trouver. Si la pièce n'a pas encore été trouvée, elle est visible sur la tableau de bord, et surmontée de flèches qui indiquent la direction dans laquelle elle se trouve. Ces flèches ne clignotent que quand un des deux joueurs a vu la pièce. Si une pièce a été ramassée, elle est grisée et n'est plus surmontée de flèches sur le tableau de bord (pièce 1 et 3 de la Figure 82). Si la pièce a été déposée dans la zone de dépose, la place qu'elle occupait dans le tableau de bord est laissée vide, afin d'assurer la stabilité d'affichage du tableau de bord. Enfin, à droite du tableau de bord, on indique le temps de jeu.
La Figure 84 présente la description d'une pièce de puzzle de RAZZLE. La pièce est d'origine numérique. Elle est identifiée dans le monde numérique par un numéro. Ce numéro est rendu perceptible sous la forme d'une image affichée dans les lunettes semi-transparentes. Son état qui peut être en attente, trouvé, ramassé ou déposé est aussi rendu perceptible dans la présentation graphique de la pièce :

- Si la pièce est en attente, elle est opaque dans le tableau de bord et affichée en superposition dans le terrain de jeu.
- Si la pièce est trouvée, les flèches dans le tableau de bord indique sa direction.
- Si la pièce est ramassée, elle est transparente dans le tableau de bord et n'est plus affichée sur le terrain de jeu.
- Si la pièce est déposée, la pièce n'apparaît plus dans le tableau de bord.

Pour présenter la pièce dans le terrain de jeu, nous prenons en compte le point de vue d'utilisateur sur la pièce. Ainsi, la position et l'orientation de l'utilisateur sont captée pour mettre à jour le point de vue sur la pièce.

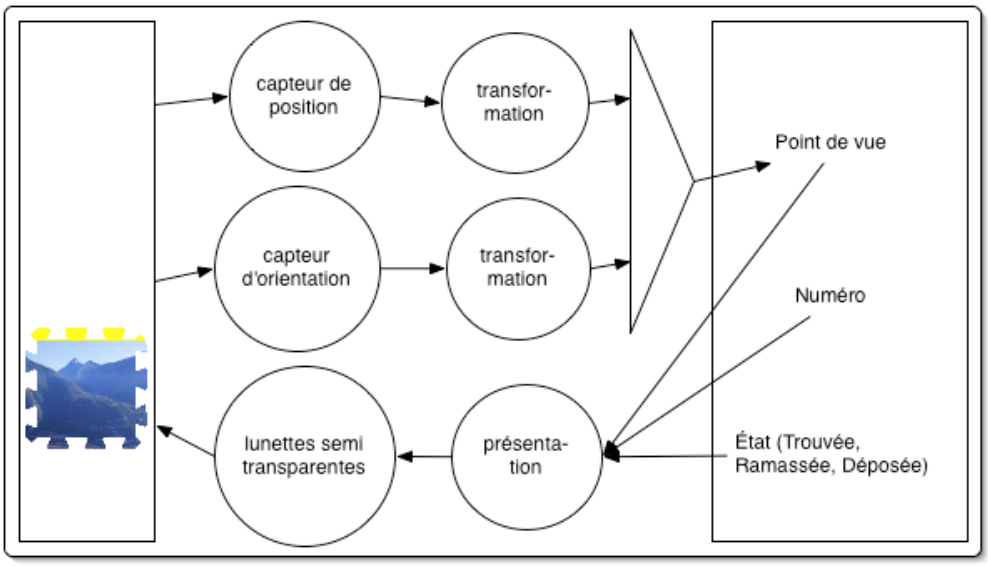


Figure 84 : Description d'une pièce de puzzle dans RAZZLE selon le modèle d'interaction mixte.

Pour ramasser les pièces de puzzle, trois techniques d'interaction sont proposées à l'utilisateur dans cette version existante de RAZZLE. Nous les présentons maintenant.

3.1.1.1.1.1. Les techniques d'interaction pour ramasser

Le joueur doit ramasser les pièces de puzzle éparpillées sur le terrain de jeu. Dans la première version existante de RAZZLE, le joueur a le choix entre trois modalités d'interaction pour ramasser les pièces.

1. La première modalité (M1) proposée est inspirée du pouvoir magique du monde des jeux vidéos. L'utilisateur s'approche de la pièce, et s'il reste suffisamment longtemps, c'est-à-dire 10 secondes dans le schéma de la Figure 85, à proximité dans un rayon de un mètre autour de la pièce, alors la pièce change d'état et est « ramassée ». La Figure 85 montre une description de cette modalité avec le modèle d'interaction mixte.

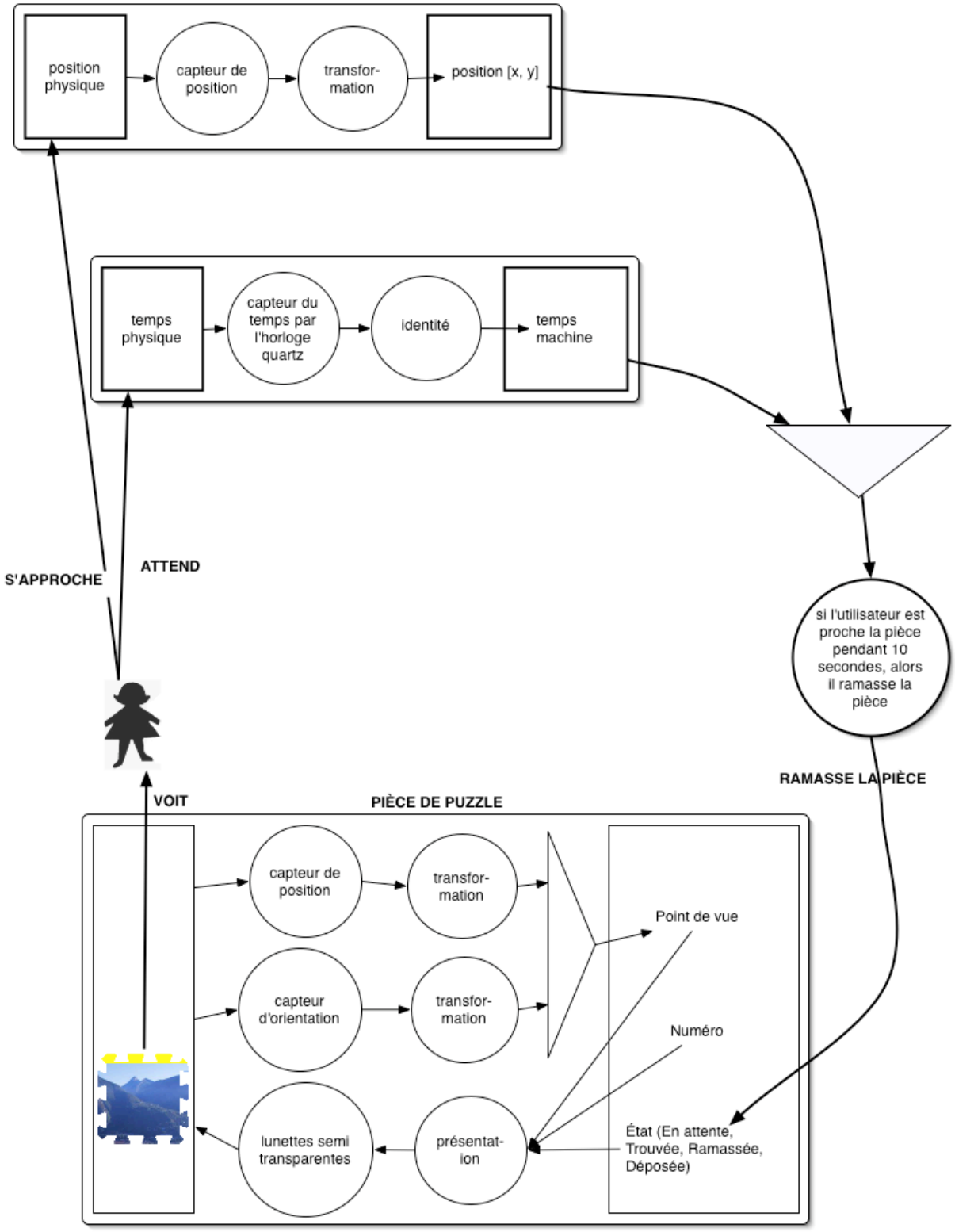


Figure 85 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE, première version : Ramasser une pièce en s'approchant à moins d'un mètre pendant quelques secondes (M1).

2. La seconde modalité (M2) proposée est inspirée du monde physique, où l'utilisateur, pour ramasser un objet, s'en approche et s'en saisit d'un geste de la main. Cette technique n'a pas été implémentée et elle est simulée par un magicien d'Oz. Cette modalité de ramassage d'une pièce mixte est décrite avec le modèle d'interaction mixte à la Figure 86.

3. La troisième modalité pour ramasser une pièce (M3) est inspirée des interfaces graphiques, où l'utilisateur, pour sélectionner un objet, amène le curseur dessus et clique avec un bouton de la souris. Dans RAZZLE, l'utilisateur, s'il voit une pièce dans son casque, peut positionner le curseur affiché dans le casque sur la pièce en manipulant la trackball, et cliquer pour la ramasser. Pour cette modalité, l'utilisateur peut opérer à distance. Cette modalité est décrite avec le modèle d'interaction mixte à la Figure 87.

Avant de commencer la conception de nouvelles modalités d'interaction, nous avons modalisé les techniques d'interaction existantes, et effectué une évaluation ergonomique du jeu afin de savoir ce qui était perfectible. Les trois schémas des figures Figure 85, Figure 86 et Figure 87 furent donc un de nos points de départ pour la conception de nouvelles techniques d'interaction pour ramasser les pièces de puzzle.

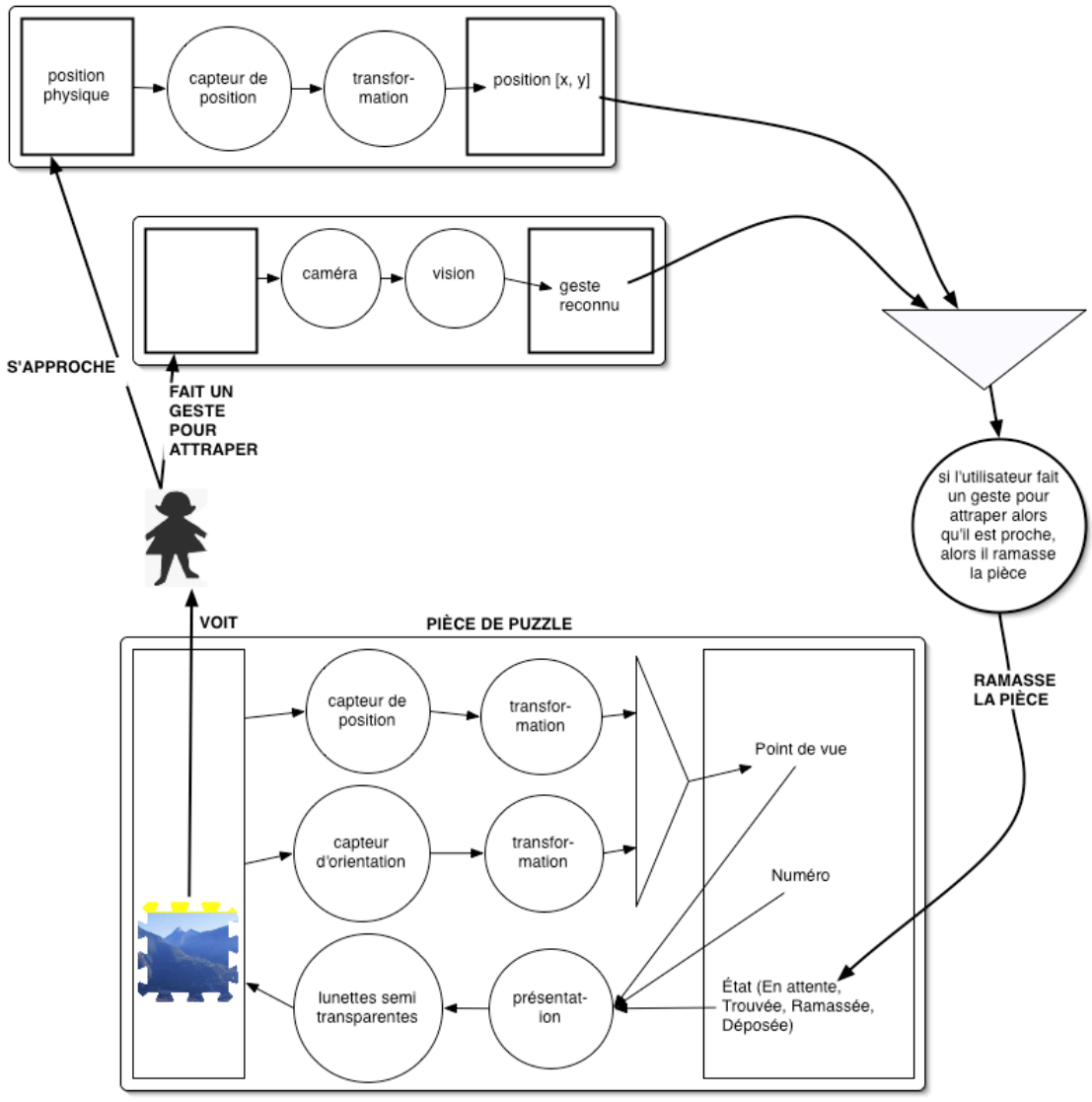


Figure 86 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE : Ramasser une pièce en s'approchant et l'attrapant d'un geste de la main (M2).

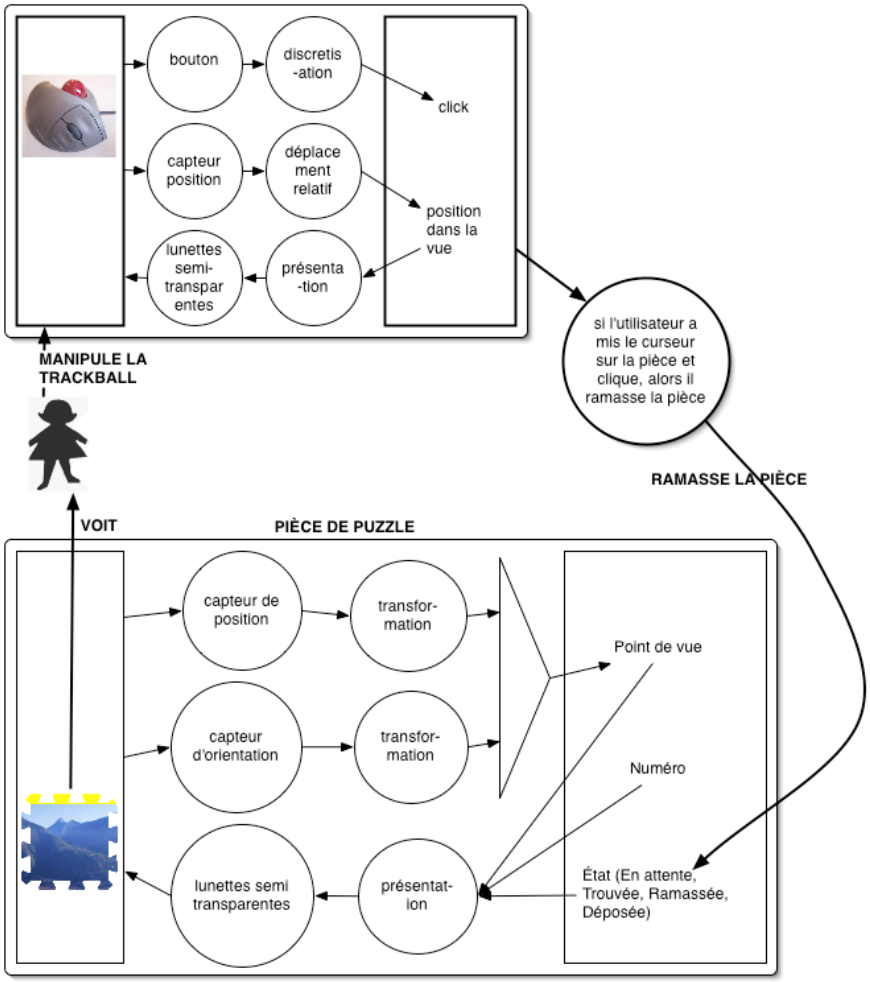


Figure 87 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE : Ramasser une pièce en positionnant le curseur de la souris dessus et en cliquant (M3).

3.1.1.1.1.2. Magicien d'Oz

RAZZLE n'a pas été conçu pour évaluer la robustesse logicielle de la localisation ou de la vision par ordinateur en contexte mobile. Par conséquent, sachant que les résultats de ces technologies sont moins bons dans ce contexte, nous les simulons pour n'étudier que l'interaction lors de l'évaluation ergonomique qui précède l'utilisation du modèle. Pour nous libérer des limites technologiques actuelles, nous avons donc opté pour le développement d'un système multimodal partiellement simulé en utilisant la technique du Magicien d'Oz. La technique d'interaction gestuelle n'est pas implémentée, ainsi que la localisation du joueur dans la salle. En effet, lors d'expériences précédentes avec MAGIC et TROC, la localisation avec un GPS avait soulevé des problèmes (comme la stabilité). La simulation est faite par un compère qui suit le joueur (Figure 83) et traduit les actions de l'utilisateur en commande système, en observant l'utilisateur. L'interface fournie au compère est présentée Figure 88.

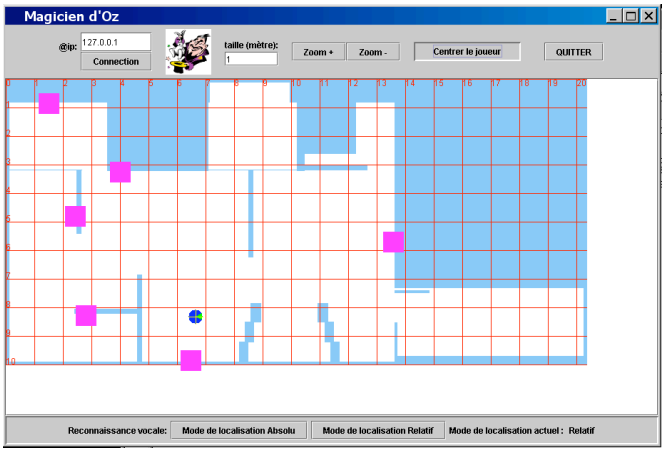


Figure 88 : Interface du magicien d'oz.

Le compère voit les gestes et la position du joueur sur le terrain de jeu. L'interface graphique du magicien d'Oz de RAZZLE (Figure 88), permet au compère de déplacer la position du joueur dans le système avec la souris et les flèches. De plus, des touches du clavier permettent de ramasser la pièce sélectionnée pour simuler la modalité M2.

3.1.1.1.1.3. Évaluation ergonomique avant re-conception

Deux phases de tests ont été définies : individuelle et par pair. Des critères ont été définis pour mener l'analyse :

- le nombre d'utilisateurs minimum pour avoir des données fiables (10),
- la façon dont les données ont été enregistrées et analysées.

Deux techniques ont été utilisées pour l'analyse des tests utilisateurs. La première, le protocole « penser tout haut[31] », consiste à demander aux utilisateurs de parler à haute voix et de formuler leurs pensées, leurs sentiments ou leurs opinions en même temps qu'ils sont en train d'accomplir leur tâche. Durant les expérimentations de RAZZLE, les utilisateurs étaient invités à parler de leurs prises de décision et de leurs difficultés lors des tests. Ceci nous a permis de détecter les problèmes avec précision. La seconde technique utilisée consistait à interroger les utilisateurs après les tests. Grâce à ces interviews, nous avons pu déterminer :

- les caractéristiques des participants (âge, sexe, catégorie socio-professionelle, expérience des
- jeux vidéos),
- l'opinion des utilisateurs sur le système.

Toutes les expérimentations ont été enregistrées à l'aide d'une caméra, et celles-ci incluaient les remarques des utilisateurs. Ainsi tout le cheminement des utilisateurs dans le terrain de jeu, et le temps passé pour accomplir la tâche ont été enregistrés. Enfin ces informations ont été analysées a posteriori grâce à une reconstitution de l'interaction par un logiciel fait sur mesure dont l'interface se présente comme celle du magicien d'Oz (Figure 88).

Les sujets pour l'expérimentation ont été recrutés suivant deux profils : pour moitié, ils jouent habituellement aux jeux vidéos en 3D (« joueurs »), et l'autre moitié n'a pas l'habitude de jouer (« non joueurs »). Pour les deux groupes de participants, c'était leur première expérience avec un système de réalité mixte. En tout, 10 utilisateurs ont pris part aux expérimentations en jouant seuls, et 14 ont joué par pair. Les utilisateurs avaient entre 18 et 40 ans, et la moyenne d'âge était de 26 ans. La proportion d'hommes était plus importante : 15 hommes parmi 24 utilisateurs. De plus, 18 utilisateurs étaient informaticiens, cette forte proportion s'expliquant par le fait qu'ils aient été recrutés parmi le personnel du laboratoire.

Pour contrôler le processus expérimental, deux personnes étaient présentes lors des tests avec les utilisateurs : un ergonome et un informaticien pour assurer le travail du compère. Tous les utilisateurs ont effectué la même tâche : chercher 4 pièces d'un même puzzle parmi les 16 pièces accrochées sur les murs du terrain de jeu, puis les déposer dans la zone de dépose. Dans les

[31] thinking aloud

expérimentations par pair, les utilisateurs avaient une contrainte supplémentaire : ils devaient respecter une limite de temps.
Les données issues des expérimentations nous ont permis de comparer les techniques d'interaction disponibles et modélisées précédemment selon le modèle d'interaction mixte. Le Tableau 16 montre comment les 40 pièces ont été ramassées pendant les expérimentations, c'est-à-dire 20 pièces par profil (« joueur » et « non joueur »).

	M1 (temporisation)	M2 (geste)	M3 (trackball)
Joueurs	12	8	0
Non joueurs	11	6	3

Tableau 16 : Utilisation globale des techniques d'interaction.

Ces chiffres montrent que la plupart des participants ont attendu près d'une pièce (M1), et que beaucoup ont choisi d'attraper la pièce avec la main (M2), au lieu de cliquer sur la pièce (M3), avec une légère différence entre les « joueurs » et les « non joueurs ». Pourtant, les commentaires des utilisateurs nous forcent à nuancer ces résultats. Ils ont expliqué qu'ils n'avaient parfois aucune intention de ramasser une pièce en attendant à proximité (M1), et qu'ils ont été surpris en s'apercevant que la pièce était déjà ramassée. Parfois ils avaient même commencé un geste pour attraper la pièce (M2), et dans ce cas, ils disent explicitement qu'ils avaient l'intention d'utiliser M2, mais qu'ils n'ont pas eu le temps. Ceci confirme une hypothèse de non observabilité qui aurait pu être faite a priori avec le modèle d'interaction mixte : la modalité M1 n'a pas de réaction (Figure 85), comme défini à la Figure 55 du Chapitre 4 (page 79).
De plus, nous remarquons que les utilisateurs ont significativement évité de se servir de la trackball pour cliquer sur les pièces (M3). En effet, ils ont expliqué que changer leur point de vue pour prendre l'outil en main et le contrôler les perturbait. Pourtant, un joueur a préféré M3, parce que cette technique d'interaction lui permettait de minimiser ses déplacements dans le terrain de jeu. Mais pour la plupart d'entre eux, M3 reste une modalité qui nécessite de changer l'objet de son attention, du terrain de jeu vers l'outil. En effet, pour utiliser la trackball, l'utilisateur doit regarder explicitement où positionner sa main, ce qui nécessite un changement de direction du regard. Ce problème aurait pu être identifié a priori sur la modélisation de la modalité M3 (Figure 87) en considérant la localisation spatiale de l'outil mixte trackball par rapport à l'objet de la tâche (caractéristique présentée au Chapitre 4 page 96). En effet, la trackball et la pièce de puzzle sont spatialement disjoints.

Nous avons constaté lors de l'évaluation ergonomique de cette première version de RAZZLE que les techniques d'interaction étaient perfectibles. Notamment, la première modalité avait été mal comprise, et la dernière nécessitait un meilleur positionnement spatial. Le premier problème était directement visible sur la modélisation (caractéristique explicite dans le modèle du Chapitre 4, Tableau 14). Le deuxième problème pouvait être vérifié sur le modèle (caractéristique indirectement exprimée du Chapitre 4, Tableau 14). Aussi suite à cette évaluation, nous avons conçu de nouvelles modalités d'interaction, en se basant sur le modèle.

3.1.1.1.2. La nouvelle version de RAZZLE : une première utilisation du modèle d'interaction mixte

Pour construire la nouvelle version de RAZZLE, nous avons travaillé avec une version alpha, incomplète, du modèle d'interaction mixte, et en collaboration avec des informaticiens spécialistes de l'Interaction Homme-Machine et des ergonomes. Pour cette utilisation du modèle d'interaction mixte, nous nous sommes concentrés sur les modalités d'interaction avec l'objet mixte. Les choix de conception n'ont pas été modifiés pour la pièce de puzzle, qui est un objet de la tâche mixte (Figure 84).
Notre objectif était d'étudier le pouvoir génératif du modèle d'interaction mixte. Nous avons choisi de concevoir des modalités d'interaction pour deux tâches : ramasser une pièce de puzzle, comme dans la version précédente de RAZZLE, mais aussi pour une nouvelles tâche : tourner la pièce de

puzzle. Ainsi dans cette version de RAZZLE, les pièces présentées dans le monde physique ne sont pas forcément bien orientées, et l'utilisateur doit les orienter correctement pour qu'elle puisse entrer dans le puzzle, avant de pouvoir les ramasser.
Nous avons conçu trois modalités, pour tourner et ramasser les pièces de puzzle. Les trois modalités sont fonctionnellement équivalentes et chacune peut être utilisée pour tourner et/ou pour ramasser une pièce. Pour ces trois modalités, il est nécessaire d'être proche de la pièce puzzle pour la tourner ou la ramasser. Nous avons conçu ces trois modalités à partir du cadre présenté à la Figure 89, en y insérant trois différentes modalités à l'intérieur.

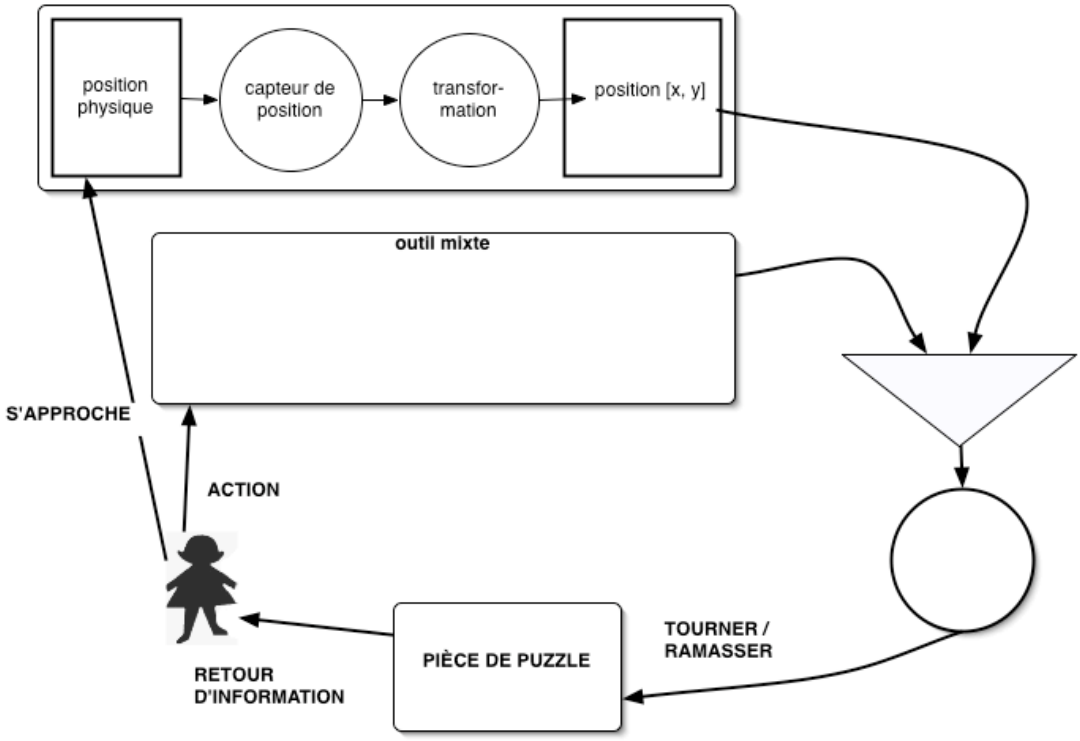


Figure 89 : Cadre pour la conception de RAZZLE.

Il s'agit d'une conception à haut niveau d'abstraction. Nous affinons ce cadre en détaillant ce cadre général pour définir l'outil mixte et le langage d'interaction. Ces deux éléments du modèle définissent une modalité d'interaction. Aussi chaque modalité est une déclinaison possible de ce cadre général. La première est la modalité tactile qui utilise le touchpad, la seconde la modalité vocale, et la troisième la gestuelle.

3.1.1.1.2.1. Modalité tactile

Pour concevoir la modalité tactile, nous avons pris comme point de départ l'évaluation ergonomique du système existant et le cadre de la Figure 89. Cette première modalité, tactile, est basée sur la modalité du trackball (Figure 87). Nous avons choisi un dispositif de liaison pour l'outil qui soit moins encombrant et qui puisse se porter au poignet grâce à un bracelet. De plus, nous avons changé les propriétés numériques de l'outil : il n'est plus nécessaire de positionner le curseur sur la pièce de puzzle, mais il suffit de faire un geste vers la gauche/droite pour tourner la pièce vers la gauche/droite et de cliquer pour ramasser la pièce de puzzle. Ainsi nous nous attendons à ce que la modalité tactile soit plus utilisée que M3. La Figure 90 présente le schéma décrivant cette modalité d'interaction modifiée et adaptée. L'utilisation du touchpad n'a d'effet que si le joueur est proche de la pièce (composition articulatoire, au niveau des outils mixtes, Figure 90). Sur un touchpad porté au poignet, l'utilisateur fait un mouvement vers la gauche ou vers la droite. Le geste vers la gauche fait tourner la pièce dans le sens antihoraire et le geste vers la droite dans le sens horaire. Pour ramasser la pièce qu'il vient d'orienter, le joueur peut cliquer sur le bouton du touchpad ou taper du doigt sur la surface.

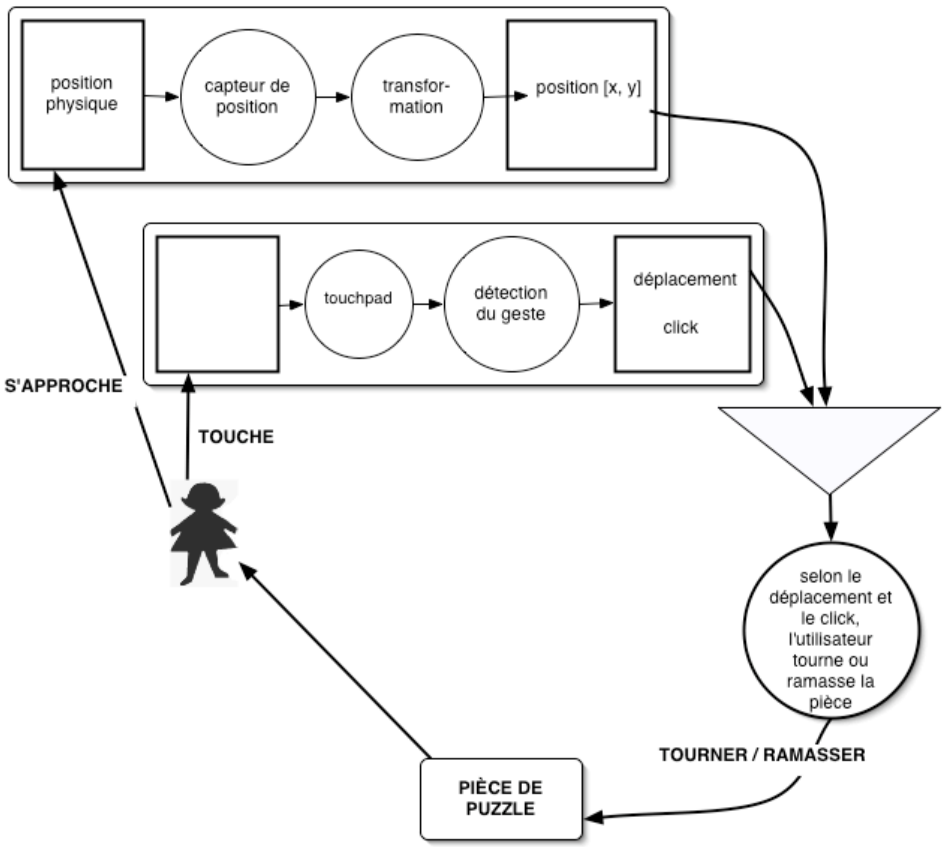


Figure 90 : Modalité tactile.

La Figure 91 présente un joueur de RAZZLE avec l'équipement nécessaire au jeu : des lunettes semi-transparentes avec un capteur d'orientation monté dessus, un touchpad, ici dans sa main, et un ordinateur portable dans le sac à dos. Il est à noter que le placement du touchpad était suggéré autour du poignet, grâce à un bracelet, mais que les utilisateurs avaient la liberté de choisir un autre emplacement.



Figure 91 : Un joueur de RAZZLE porte l'équipement nécessaire (des lunettes semi-transparentes avec un capteur d'orientation monté dessus, un touchpad, ici dans sa main, et un ordinateur portable dans le sac à dos).

3.1.1.1.2.2. Modalité vocale

Pour concevoir la modalité vocale, nous avons aussi pris comme point de départ l'évaluation ergonomique du système existant et le cadre de la Figure 89. Voyant que la modalité M1 (temporisation, Figure 85) n'avait pas été bien utilisée par les utilisateurs de la première version de RAZZLE, nous avons re-conçu cette modalité pour qu'elle permette toujours de ramasser une pièce de puzzle sans bouger la main, mais tout de même en demandant une action nécessairement explicite de l'utilisateur. Nous avons alors pensé à une modalité de liaison (micro, reconnaissance vocale) pour l'outil. La Figure 92 présente le schéma décrivant cette nouvelle modalité d'interaction pour tourner et ramasser les pièces de puzzle. L'utilisation de la modalité vocale n'a d'effet que si le joueur est proche de la pièce. À l'aide du microphone, le joueur dit qu'il veut faire tourner la pièce en précisant le sens de rotation voulu. Étaient autorisées les commandes vocales telles que « vers la gauche », « gauche », « tourner à gauche », etc. Pour ramasser, l'utilisateur peut signifier au système en disant des mots tels que « attrape » , « ramasser », « je prends », etc. qu'il veut ramasser la pièce de puzzle.

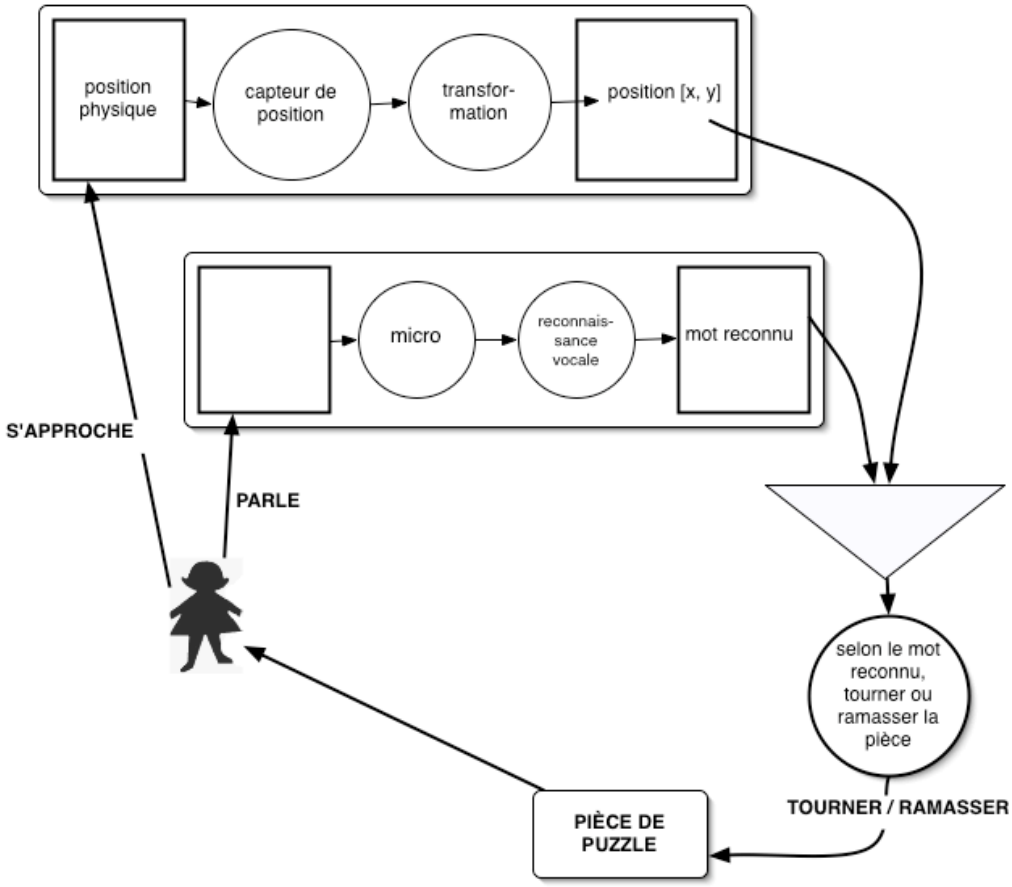


Figure 92 : Modalité vocale.

3.1.1.1.2.3. Modalité gestuelle

Pour concevoir la modalité gestuelle, nous avons aussi pris comme point de départ l'évaluation ergonomique du système existant et le cadre de la Figure 89. Comme la modalité M2 (geste, Figure 86) avait bien été utilisée et comprise, nous l'avons conservée. À l'aide de sa main dirigée vers la pièce sélectionnée, l'utilisateur esquisse un geste dans le sens voulu de rotation, afin de faire tourner la pièce. Et en faisant un geste de saisie dirigée vers la pièce sélectionnée, l'utilisateur ramasse la pièce. De même que pour les autres modalités, l'utilisation de la modalité gestuelle n'a d'effet que si le joueur est proche de la pièce. La modalité gestuelle est représentée Figure 93. La Figure 94 présente un utilisateur se servant de cette modalité pour ramasser une pièce de puzzle.

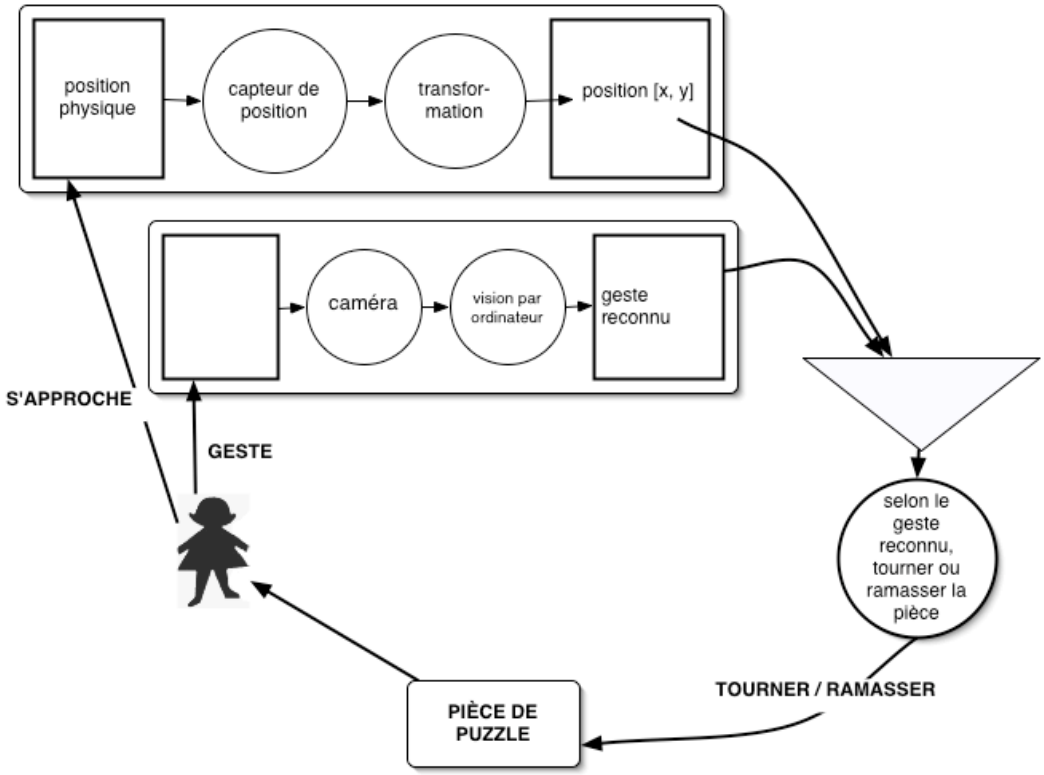


Figure 93 : Modalité gestuelle.

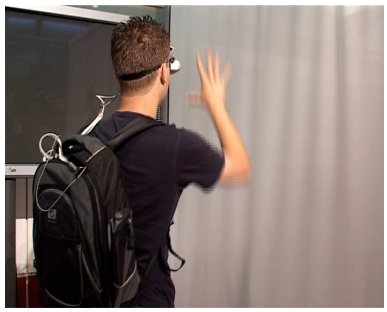


Figure 94 : Un utilisateur ramasse une pièce trouvée grâce à la modalité gestuelle.

Nous avons ainsi conçu trois modalités d'interaction en nous aidant du cadre défini par le modèle d'interaction mixte. Nous avons pris comme point de départ ce cadre, ainsi que la première évaluation ergonomique de RAZZLE. Notre objectif était d'évaluer le pouvoir génératif du modèle d'interaction mixte dans une situation réelle, et voir si le modèle nous permettrait effectivement de concevoir une meilleure solution pour ce système particulier.
Pour vérifier nos objectifs, les trois modalités pour tourner puis ramasser les pièces de puzzle ont été implémentées afin d'en observer l'usage par des utilisateurs mobiles. Pourtant, pour les mêmes raisons que pour la première version de RAZZLE (limites technologiques), certaines parties du logiciel ne sont pas implémentées, mais simulées grâce à la technique du magicien d'Oz : c'est le cas de la modalité vocale et gestuelle. Nous avons ensuite souhaité effectuer une nouvelle évaluation ergonomique de cette version de RAZZLE afin de mettre à l'épreuve nos objectifs avec la nouvelle version de RAZZLE conçue avec le modèle d'interaction mixte.

3.1.1.1.3. Évaluation ergonomique de la nouvelle version conçue

Cette nouvelle version du système a fait l'objet d'une évaluation ergonomique [Coutrix et al., 2006]. Cette expérience visait à étudier l'utilisation des différentes modalités conçues avec le modèle d'interaction mixte. Comme dans l'expérience précédente, nous avons simulé les techniques d'interaction vocale et gestuelle, ainsi que la position du joueur. Pour cette évaluation, le magicien

était caché dans une salle à part et entendait dans le casque ce que disait le joueur dans le micro. Il observait également la position du joueur grâce aux caméras.

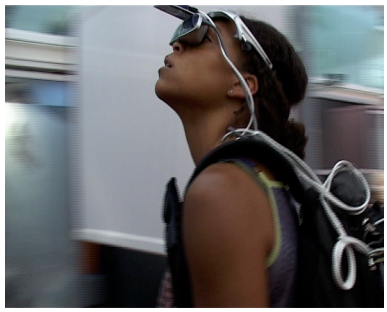


Figure 95 : Une joueuse est à la recherche des pièces à trouver pendant l'évaluation ergonomique de RAZZLE.

Les sujets qui ont participé à cette expérience étaient au nombre de 10 et avaient entre 20 et 30 ans. Comme nous n'étudions pas l'influence des jeux vidéos traditionnels pour l'étude de ce jeu, nous avons choisi des sujets ayant peu d'expérience dans les jeux vidéos. Ils ont été convoqués pour trois sessions de jeu. Les trois sessions expérimentales ont été réalisées sur 6 jours. Les pièces de puzzle étaient réparties différemment sur le terrain de jeu pour chaque session. L'activité des sujets durant les sessions (d'une durée moyenne de 10 minutes) était enregistrée au moyen de différents capteurs : caméras, capteurs d'orientation, traces d'événements systèmes. Les sessions étaient immédiatement suivies de séances d'entretiens semi directifs qui faisaient l'objet d'un enregistrement vidéo.
Les actions des sujets et les modalités utilisées ont été recueillies au cours de l'étude à partir de film vidéo. De plus, un enregistrement des entretiens a été réalisé afin de permettre des analyses qualitatives à partir de l'identification des attentes, des intentions et des stratégies des sujets quant à l'usage des modalités.
À partir des données ainsi recueillies, nous avons procédé à leur dépouillement et analyse. Nous en présentons les résultats : les premiers résultats quant à l'usage des trois modalités d'interaction viennent de l'étude de l'ensemble des utilisateurs dans sa globalité. Puis dans un second temps, d'autres traits, individuels, seront mis en relief.

3.1.1.1.3.1. Utilisation globale des modalités d'interaction

Un des premiers résultats obtenus sur l'utilisation des modalités montre que toutes les modalités ont été utilisées pour toutes les sessions et sujets confondus. Mais nous pouvons noter dans la Figure 96 une préférence d'utilisation pour la modalité vocale (44,97%) par rapport aux utilisations des modalités gestuelle (30,67%) et tactile (24,34%).

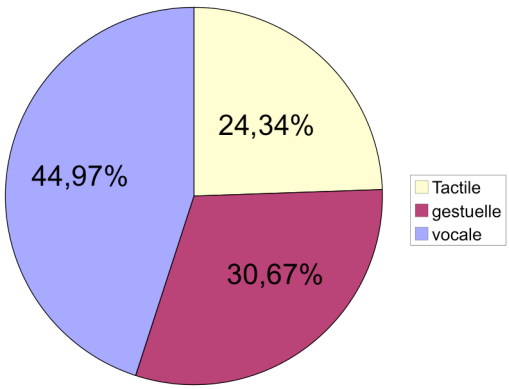


Figure 96 : Taux d'utilisation des modalités pour toutes sessions et sujets confondus.

Si nous analysons l'utilisation des modalités toutes sessions confondues pour chaque sujet (Figure 97), les résultats indiquent également que l'ensemble des modalités a été utilisé par tous les sujets sauf par un sujet (sujet 2 à la Figure 97).

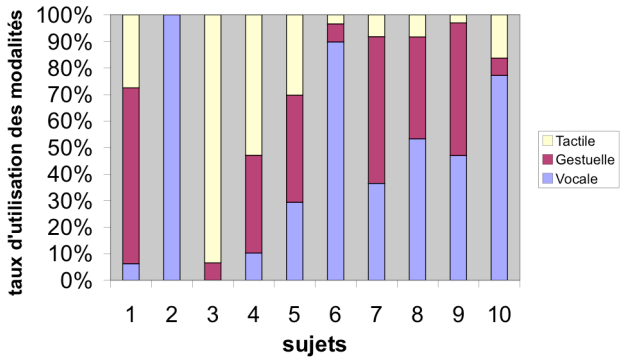


Figure 97 : Taux d'utilisation des modalités par sujet (toutes commandes et sessions confondues).

Par ailleurs, d'une session à l'autre, nous observons une évolution dans l'utilisation des modalités :

- Lors des premières sessions, nous observons pour tous les sujets (sauf un), une répartition d'utilisation d'au moins deux modalités, ce qui correspond à une phase de test informel exploratoire du système. Ce fait est corroboré par les données issues des auto confrontations.
- Dans les sessions suivantes, des préférences dans l'utilisation des modalités apparaissent. Ces préférences varient en fonction des sujets. La Figure 98 présente les modalités utilisées par sujet à la première session et à la troisième session. Entre les deux sessions, les sujets ont développé une préférence pour une modalité particulière. Par exemple, le sujet 1, qui avait utilisé les modalités tactile et gestuelle à la première session, n'utilise plus que la modalité gestuelle à la troisième session.

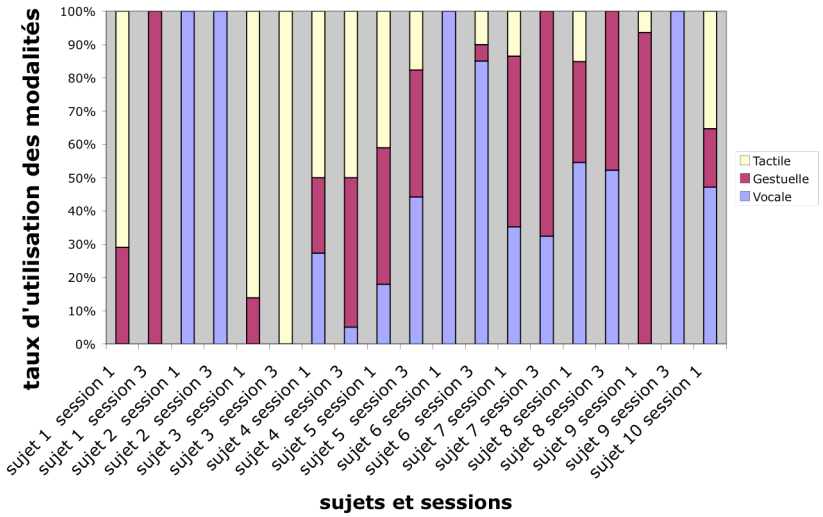


Figure 98 : Comparaison de l'usage des modalités par sujet entre la première et la troisième session (toutes actions confondues).

De plus, si nous considérons l'ensemble des sessions, nous observons la confirmation des résultats globaux à savoir la légère préférence pour une modalité commune qui serait la modalité vocale à l'ensemble des sujets pour effectuer l'une ou l'autre des actions (spécialisation remarquable dans la Figure 99).

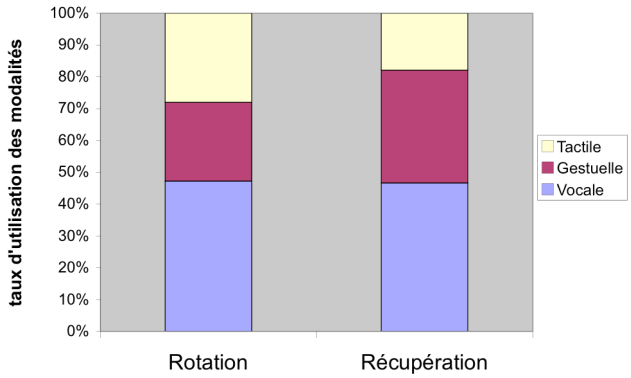


Figure 99 : Taux d'utilisation des modalités pour une action donnée (toutes sessions et sujets confondus).

En affinant les résultats et en croisant chacune des actions avec chacune des sessions comme dans la Figure 100, nous n'observons toujours pas la domination d'une autre modalité commune à l'ensemble des sujets pour effectuer l'une ou l'autre des actions. Par conséquent, s'il y a spécialisation, elle ne peut se situer qu'à un niveau individuel, ce qui nous a conduit à effectuer une analyse spécifique des données pour chaque sujet.

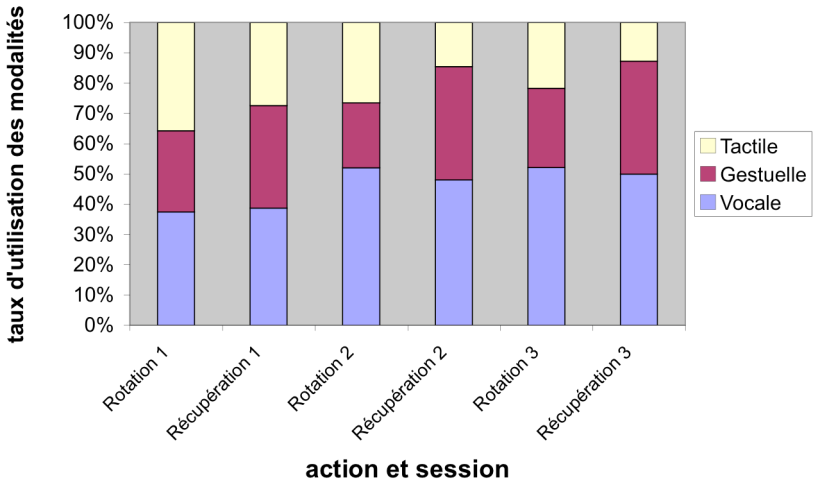


Figure 100 : Taux d'utilisation des modalités pour chaque action et chaque session.

3.1.1.1.3.2. Spécialisation intra-individuelle

Comme mentionné ci-dessus, nous observons chez de nombreux sujets une tendance à utiliser préférentiellement une modalité qui peut-être associée à une action ou non. Ces tendances peuvent être :

- Ponctuelles : elles n'apparaissent qu'au cours d'une seule session d'interaction ;
- Récurrentes : elles apparaissent dans d'autres sessions.

Elles peuvent aussi être individualisées ou plurielles. Une utilisation individualisée correspond au cas où une même modalité est utilisée pour réaliser une seule et même commande. Dans la Figure 101, nous avons extrait les résultats du sujet 4 pour montrer un exemple de spécialisation où plusieurs modalités sont utilisées, chacune assignée par l'utilisateur à une commande. Ainsi à la Figure 101 nous observons une spécialisation rapide de la modalité tactile avec le touchpad pour effectuer les rotations et une utilisation récurrente de la modalité gestuelle pour l'action de récupération de la pièce.

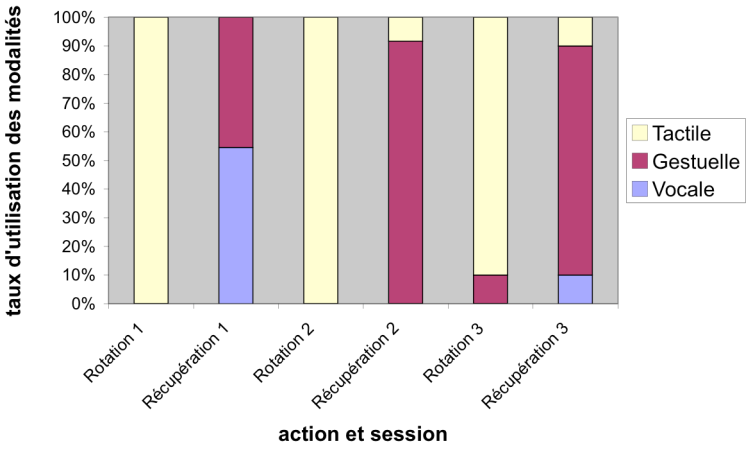


Figure 101 : Taux d'utilisation des modalités pour chaque action et session par le sujet 4.

Nous remarquons aussi qu'une même modalité peut être utilisée pour réaliser plusieurs tâches (rotation et récupération). Dans la Figure 102, nous avons extrait les résultats du sujet 9 pour montrer un exemple de spécialisation d'une modalité quelle que soit l'action à réaliser. Nous pouvons remarquer une spécialisation rapide de la modalité vocale pour effectuer l'une ou l'autre des actions de rotation et de récupération.

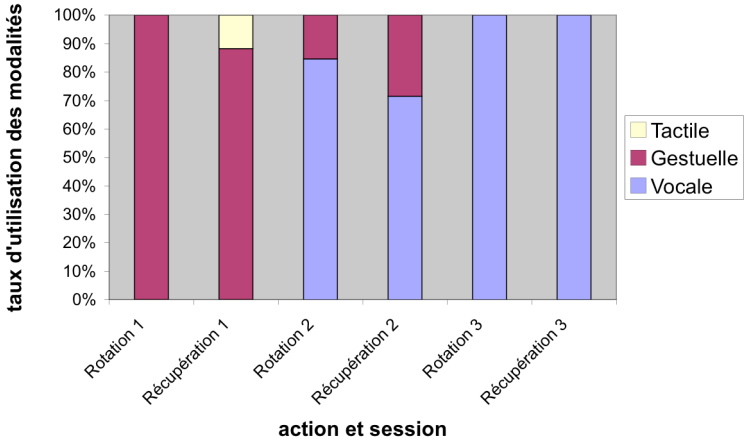


Figure 102 : Taux d'utilisation des modalités pour chaque action et session par le sujet 9.

3.1.1.1.4. Synthèse de l'expérience de conception

En résumé, même si la première version du modèle d'interaction mixte utilisée pour concevoir les techniques d'interaction de la nouvelle version de RAZZLE nous a permis d'explorer l'espace de conception de façon très limitée, ces résultats sont d'ores et déjà encourageants. En effet, nous avons grâce au cadre de conception pu imaginer comment améliorer les techniques d'interaction. De plus, nous avons fait des hypothèses sur l'utilisation de ces nouvelles modalités, connaissant les résultats de l'évaluation ergonomique précédente. Les résultats de l'expérience suivante a des implications pour le pouvoir génératif du modèle d'interaction mixte : tout d'abord, ils ont confirmé les hypothèses faites. Nous pensions par exemple que la modalité tactile serait plus utilisée et elle a effectivement plus utilisée. Ensuite, les expériences nous ont conforté dans l'idée de décrire plusieurs paradigmes d'interactions, puisque nous avons vu que les modalités d'interaction (vocale, gestuelle) ou l'outil (touchpad) sont tous utilisés, bien que n'appartenant pas au même paradigme d'interaction. L'aspect intégrateur du modèle a permis cela. Nous pensons qu'il est important de ne pas exclure un paradigme qui est utilisé. Enfin, il nous a paru important à l'issue de cette expérience de développer le nombre de caractéristiques pertinentes dans l'espace de conception. En effet,

l'évaluation confirme que les alternatives de conception sont utiles puisque toutes les modalités ont été utilisées et n'auraient pas pu être écartées.
En résumé, si la première version du modèle d'interaction mixte a un pouvoir génératif encourageant, il reste néanmoins limité à quelques dimensions (composition spatiale des objets entre eux (Chapitre 4, section 2.2.4), présence ou absence d'une modalité de liaison en sortie (Chapitre 4, section 2.1.3.1), et nous avons souhaité l'étendre pour les expériences suivantes avec d'autres caractéristiques. Nous avons donc intégré des caractéristiques des modalités de liaison. Nous présentons maintenant l'expérience suivante, sur Playground, une installation artistique interactive.

3.1.1.2. Playground : une installation artistique interactive

Playground (Figure 103) est une installation artistique interactive que nous avons conçue en février 2007. Dans la salle où *Playground* est installée se trouve un parc pour enfant. Un fond sonore de cour de récréation paisible est diffusé à partir de haut-parleurs dissimulés dans les objets jonchant le sol du parc. La porte du parc est ouverte, invitant le spectateur à entrer. À l'intérieur, il n'y a de la place que pour une personne, l'exposant au regard des autres visiteurs de l'exposition. Le seuil franchi, le visiteur marche sur un tapis qui détecte sa présence, et des cris et des pleurs se font entendre. Lorsque le visiteur sort, le son revient progressivement à son état paisible.



Figure 103 : Vue de l'installation interactive *Playground* à la Gallerie DuBellay, Mont Saint-Aignan, Seine-Maritime, exposée en mars 2007 lors de l'exposition intitulée *Groupe*.

Le tapis, détecteur de présence, a été construit de façon ad-hoc : il se constitue de plaques de carton, séparées par de la mousse percée de trous. Au travers des trous dans la mousse, le contact entre les plaques se fait lorsque le visiteur marche dessus. Les plaques de carton sont entourées de papier en aluminium relié aux composants matériels du bouton gauche d'une souris branchée à un ordinateur. Les évènements « Mouse pressed » et « Mouse released » sont ensuite simplement pris en compte par un applet java qui gère l'interaction. La Figure 104 présente la solution finale retenue pour l'installation, décrite avec le modèle d'interaction mixte. Parmi les propriétés physiques du parc de *Playground*, le tapis et son langage de liaison permettent de détecter la présence d'un visiteur dans le parc. Une propriété numérique indique si une présence a été détectée, une autre propriété numérique contient les sons qui sont diffusés par les haut-parleurs. Un langage de liaison en sortie permet de diffuser le son adéquat, selon qu'une présence a été détectée ou non. Nous attirons l'attention du lecteur sur le fait que dans le cas de cette installation artistique, il s'agissait seulement de concevoir un objet mixte, mais qu'il n'y a pas de tâche à proprement parler.

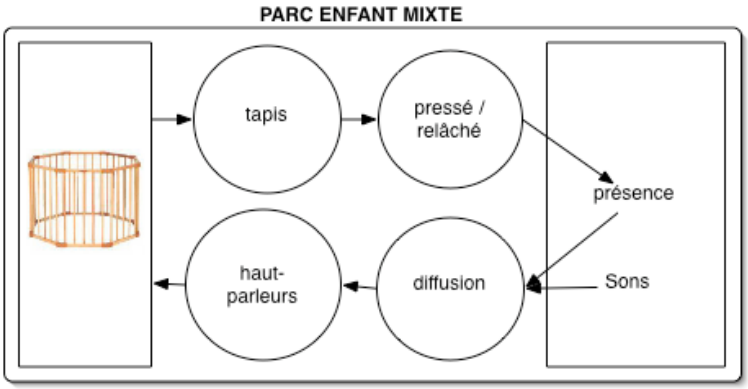


Figure 104 : Schéma décrivant le parc augmenté selon le modèle d'interaction mixte.

Nous avons, grâce au modèle d'interaction mixte, conçu différente solution pour cette installation interactive. Tout d'abord, pour détecter la présence, nous avons imaginé d'autres modalités de liaison : (camera, vision par ordinateur), (capteur de mouvement, dépassement d'une certaine valeur seuil) ou (proximètre infrarouge, dépassement d'une certaine valeur seuil). Toutes ces modalités de liaison permettent de savoir si quelqu'un est dans le parc. De même, pour la modalité de liaison en sortie, nous avons imaginé plusieurs solutions : des projections d'images, du son spatialisé, etc. Après cette phase d'exploration des possibilités, nous avons finalement choisi les modalités dont le dispositif de liaison était le moins cher (financièrement) et qui pouvait nous garantir l'effet souhaité : un tapis de présence fabriqué à partir d'une souris démontée pour l'entrée et des haut-parleurs pour la sortie.
Pendant cette expérience de conception d'une installation artistique interactive, nous avons un peu plus appuyé notre réflexion sur le cadre fourni par le modèle d'interaction mixte. Il nous a permis d'explorer diverses solutions répondant à nos attentes en raisonnant sur les deux niveaux d'abstraction (dispositif, langage) des modalités de liaisons. Ceci montre un certain pouvoir génératif du modèle et souligne aussi que le modèle ne définit pas de contraintes trop fortes pour le cas de la création d'une installation artistique, qui répond à des critères différents des interfaces traditionnelles.
Nous avons conduit une troisième étude du modèle pour un système plus classique en informatique qui est un jeu de réalité mixte et mobile. Tout comme les deux premières expérimentations, les utilisateurs du modèle sont des experts.

3.1.1.3. Snap2Play : un jeu de Memory mobile et mixte

Nous avons utilisé le modèle d'interaction mixte pour concevoir l'interaction pour un jeu de Memory en réalité augmentée mobile. Le jeu, baptisé Snap2Play, a été imaginé à partir d'une première application Snap2Tell [Chevallet et al., 2005][Chevallet et al., 2007] de nos collaborateurs en recherche d'information du projet MOSAIC (http://ipal.i2r.a-star.edu.sg/Projects/Mosaic/index.html). Nous avons conçu les cartes de Memory augmentées ainsi que les techniques d'interaction pour les ramasser.

Snap2Play [Chin et al., 2008a][Chin et al., 2008b][You et al., 2008] est inspiré du Memory. Le Memory est un jeu de cartes constitué d'un ensemble de paires de cartes identiques. Toutes les cartes sont retournées sur la table. Deux cartes sont découvertes à chaque tour. Le but du jeu est de retourner les paires de cartes identiques. Dans Snap2Play, le joueur doit retrouver les cartes identiques non pas sur la table, mais le long d'un parcours prédéfini en extérieur (Figure 105). Le but du jeu est de reconstituer les paires de cartes, dont l'une est physique et l'autre numérique. Pour terminer le jeu, les joueurs doivent reconstituer 3 paires de cartes, disposées en séquence sur le parcours, et commençant par la carte numérique suivie de la carte physique.

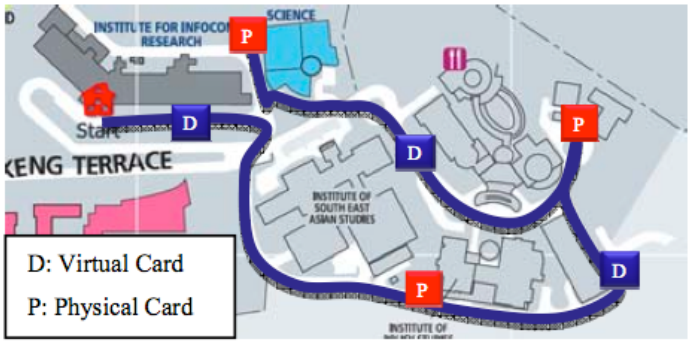


Figure 105: Un exemple de parcours du jeu Snap2Play sur le campus de la National University Singapore. Les points bleus (D) sur le parcours indique une carte numérique, et les points (P) indique une carte physique.

Le jeu se joue sur téléphone portable Nokia N80, accompagné d'un récepteur GPS et d'un dispositif SHAKE (http://www.samh-engineering.com/) contenant des composants matériels dont des accéléromètres. La Figure 106 présente une succession de capture d'écran montrant l'interface du jeu en sortie. L'utilisateur collecte une carte numérique (Figure 106, écrans 1 à 5) puis la carte physique correspondante (Figure 106, écrans 6 à 8). Les écrans 1 et 2 montrent l'indice pour trouver la carte numérique sur le terrain de jeu. Sur l'écran 3, l'utilisateur est suffisamment proche de la pièce numérique. On affiche alors une vue radar du terrain de jeu pour que l'utilisateur puisse trouver l'emplacement exact de la pièce. Une fois atteint l'emplacement de la pièce, l'utilisateur doit trouver sa position en orientant le téléphone dans toutes les directions jusqu'à trouver l'objet présenté sur l'écran 4 (gauche) de la Figure 106. Une fois cette carte collectée, le système affiche un message « *You have found the token* » comme le présente l'écran 4 (droite) de la Figure 106. On lui présente alors la carte numérique de la paire (écran 5). Il doit alors aller collecter la deuxième carte de la paire qui est un bâtiment physique (carte physique). L'écran 6 montre la vue radar présentée à l'utilisateur proche de la deuxième carte, et l'écran 7 montre la 2ème carte (physique), c'est-à-dire la scène physique vue à travers le téléphone. Une fois collectée (écran 8), le joueur est invité à continuer le jeu avec une deuxième paire à trouver.



Figure 106 : Captures d'écrans du téléphone portable dans les différentes phases du jeu Snap2Play.

Grâce au modèle d'interaction mixte, nous avons conçu des techniques d'interaction pour collecter les cartes numériques et physiques. Lors de cette utilisation du modèle d'interaction mixte, notre objectif était de fournir des techniques d'interaction similaires entre monde numérique et monde physique.

3.1.1.3.1. Objet de la tâche : la carte de Memory

Nous avons commencé par concevoir des cartes physiques et numériques, en cherchant à les faire aussi similaires que possibles du point de vue de l'utilisateur. Les étapes suivies pour la conception, basées sur notre modèle, sont présentées de la Figure 107 à la Figure 113. Au départ (Figure 107), nous connaissions les scènes physiques choisis comme cartes physiques (Figure 107, haut) et les propriétés numériques des cartes numériques correspondantes (Figure 107, bas) :

- La position numérique, issue des relevés GPS préalables,
- l'orientation de la carte, décidée au préalable,
- la distance entre l'utilisateur et la carte, qui sera utile pour savoir si la carte est observable et/ou peut être ramassée,
- les images de référence de la scène physique correspondante.

Les deux autres propriétés numériques sont des indicateurs pour connaître l'état de la carte de Memory (est présentée, est ramassée).

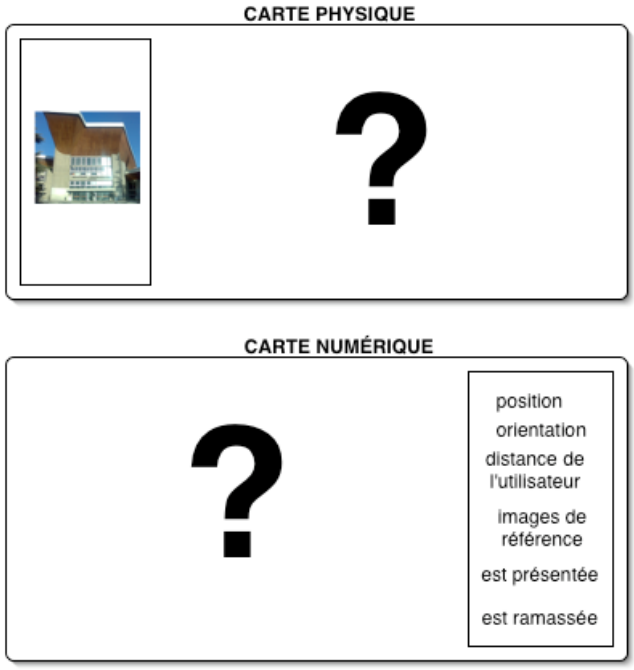


Figure 107 : Point de départ pour concevoir des cartes de jeu de Memory similaires dans Snap2Play.

Afin de concevoir une interaction similaire avec ces deux types de cartes, nous avons copié des propriétés physiques de la carte physique pour construire la carte numérique, et nous avons copié des propriétés numériques de la carte numérique pour construire la carte physique (Figure 108).

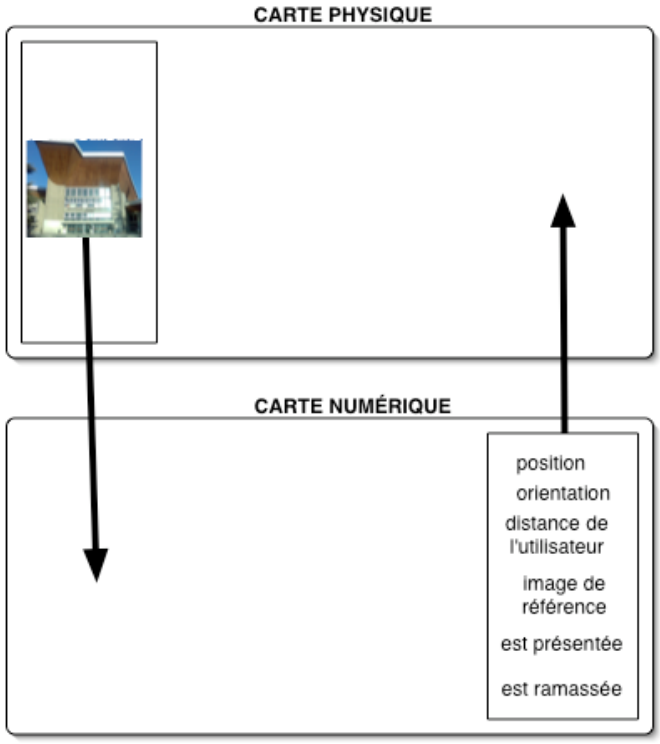


Figure 108 : Copier les propriétés physiques et numériques pour concevoir des cartes de jeu de Memory similaires dans Snap2Play.

La Figure 109 présente les propriétés que nous avons choisies de copier. À partir des propriétés physiques de la carte physique, nous choisissons de doter la carte numérique d'une image physique, ainsi que d'une position et d'une orientation dans l'espace physique (Figure 109, du haut vers le bas). À partir des propriétés numériques de la carte numérique, nous choisissons de doter la carte physique d'une position numérique, relevé GPS de sa position dans l'espace physique, d'une propriété indiquant sa distance à l'utilisateur, d'un ensemble d'images numériques de référence de cette carte physique (Figure 109, du bas vers le haut). Nous copions également la propriété numérique indiquant si la pièce a été ramassée, mais pas celle indiquant si elle est présentée, puisque dans le cas de la carte physique, cette propriété n'est pas utile.

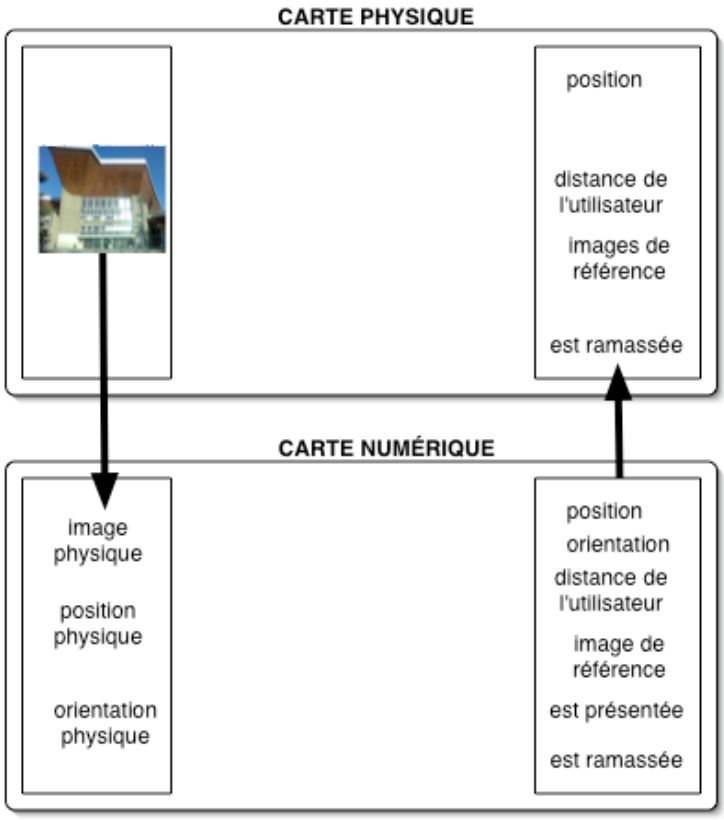


Figure 109 : Copier les propriétés physiques et numériques nécessaires pour concevoir des cartes de jeu de Memory similaires dans Snap2Play.

L'étape suivante de conception (Figure 110) consiste à définir des modalités de liaison pour lier ces propriétés. Par exemple, il nous faut lier la position physique avec la position numérique des cartes. Pour cela, nous avons le choix entre plusieurs modalités de liaison : utiliser le GPS, le réseau GSM, ou encore l'intensité des signaux de réseaux sans fils. Nous avons choisi le GPS pour des raisons de fiabilité.

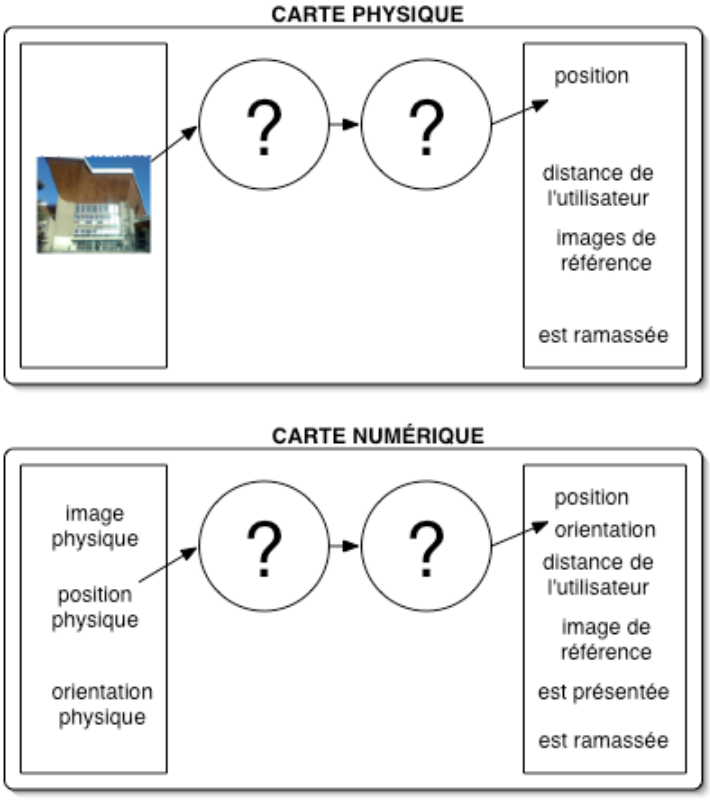


Figure 110 : Concevoir une modalité de liaison en entrée pour les cartes de jeu de Memory dans Snap2Play.

Dans l'autre sens (Figure 111), nous pouvons définir une modalité de liaison en sortie pour matérialiser la propriété numérique « est ramassée ». Nous avons à notre disposition sur le téléphone deux dispositifs : l'écran et le haut-parleur (Figure 111). Plusieurs langages de liaison peuvent être utilisés. Pour le haut-parleur, nous pouvons imaginer un son arbitraire, ou la synthèse de la parole (par exemple le mot « *Collected* » dit) (Figure 111). Pour l'écran, nous pouvons imaginer d'afficher un signe « ✔ » ou un message en langue naturelle « *You have found the token* » (Figure 111).
Nous avons choisi le message le plus discret pour un lieu public (pas de son) et le plus explicite (« *You have found the token* » est plus explicite que « ✔ »).

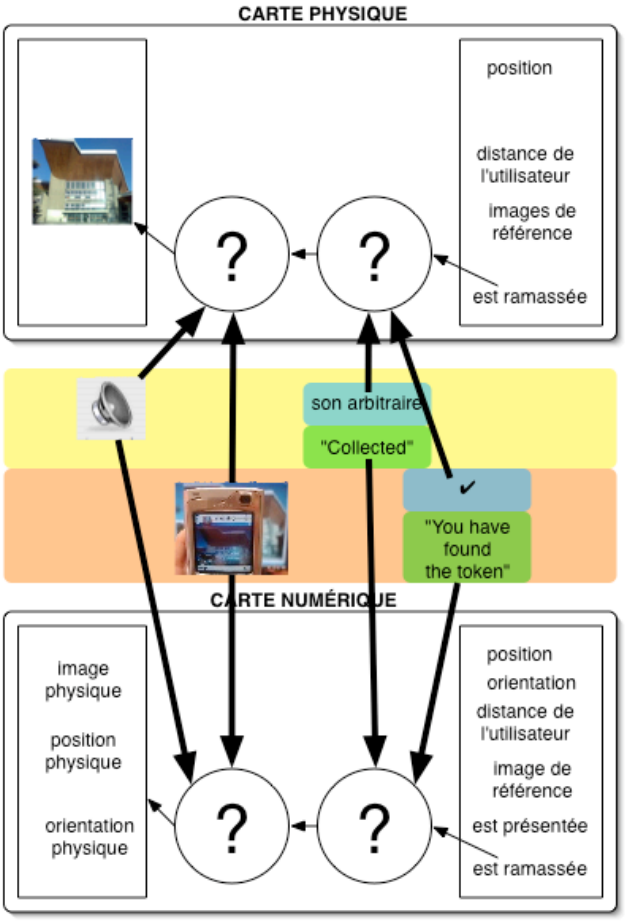


Figure 111 : Concevoir une modalité de liaison en sortie pour les cartes de jeu de Memory dans Snap2Play : rendre observable l'état de la carte.

De plus, pour la carte numérique, il nous faut définir une modalité de liaison en sortie afin de rendre observable l'image de référence de la carte (Figure 112). Pour cela nous choisissons un langage de liaison graphique. Nous pouvions utiliser des lunettes semi transparentes, l'écran du téléphone, ou encore un mini projecteur. Les lunettes ou le projecteur pourraient accentuer l'impression d'ancrage de la carte numérique dans le monde physique. Néanmoins, afin de ne pas encombrer l'utilisateur dans un jeu qui est conçu pour être jouable sur téléphone mobile, nous avons choisi l'écran intégré au téléphone.
La Figure 113 montre la conception des cartes physiques et numériques augmentées.

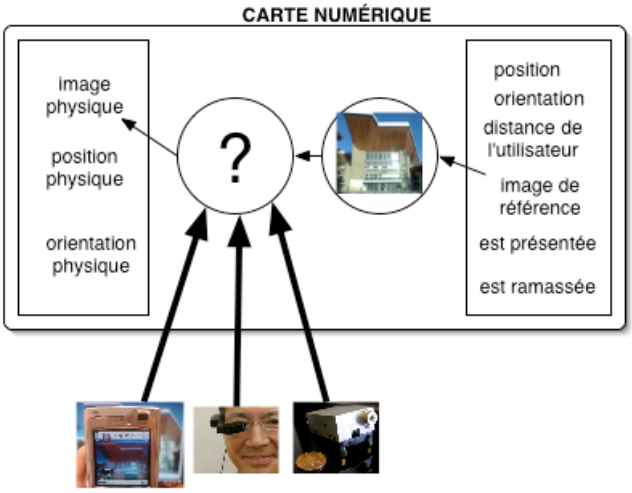


Figure 112 : Concevoir une modalité de liaison en sortie pour les cartes de jeu de Memory dans Snap2Play : rendre observable l'image de référence de la carte.

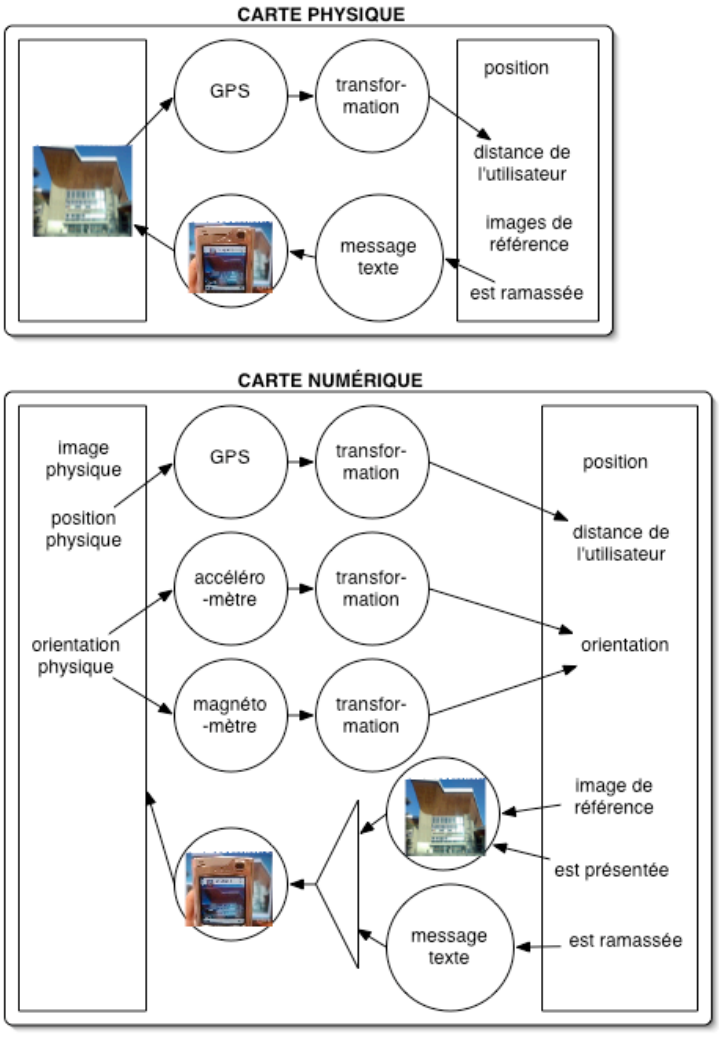


Figure 113 : Conception finale retenue pour les cartes de jeu de Memory dans Snap2Play.

L'utilisation du modèle d'interaction mixte pour explorer l'espace de conception a été beaucoup plus explicite et systématique dans cet exemple de conception que dans l'expérience avec le jeu RAZZLE ou l'installation *Playground*. De ce point de vue, nous avons été un peu plus loin dans l'évaluation du pouvoir génératif du modèle d'interaction mixte.

Une fois la conception de l'objet de la tâche terminée, nous avons conçu des techniques d'interaction pour ramasser les cartes de Memory. Nous présentons maintenant comment nous avons exploré l'espace de conception en nous reposant sur le modèle et comment nous avons cherché à fournir des techniques d'interaction similaires pour les cartes physiques et numériques.

3.1.1.3.2. Technique d'interaction pour collecter l'objet de la tâche

Grâce au modèle d'interaction mixte, nous avons pu explorer les possibilités de techniques d'interaction pour changer l'état de la propriété numérique des cartes « est ramassée ». En nous inspirant des techniques d'interaction validées dans RAZZLE, nous avons proposé que l'utilisateur se serve de la modalité d'interaction gestuelle (comme pour RAZZLE, Figure 94 page 127) : l'utilisateur positionne son téléphone pour voir la carte sur l'écran et fait un geste vers l'image physique de la carte pour ramasser. Nous avons aussi pensé à la parole (Figure 114) : l'utilisateur positionne son téléphone pour voir la carte sur l'écran et dit « *collect*[32] ». Nous avons enfin envisagé une modalité utilisant le bouton central du téléphone (Figure 115) : l'utilisateur positionne son téléphone pour voir l'image de la carte sur l'écran et clique sur le bouton pour ramasser la carte.



Figure 114 : Ramasser une carte de Memory par la parole.



Figure 115 : Ramasser une carte de Memory en « prenant une photo » : il s'agit de viser et appuyer sur un bouton.

Pour des questions de faisabilité et robustesse logicielle en contexte mobile, nous avons choisi la modalité prenant un compte le click sur le bouton central du téléphone. La Figure 116 présente le dispositif d'interaction composé de deux outils pour prendre les cartes de Memory. Le premier est le bouton (Figure 116, haut), le deuxième est la caméra (Figure 116, bas). Les deux outils sont composés au niveau articulatoire, avant le langage d'interaction : le langage d'interaction a besoin des informations du bouton (« est cliqué ») et de la caméra (« image ») pour pouvoir appliquer la tâche élémentaire « ramasser » à la carte de Memory. Pour faire en sorte que les modalités d'interaction soient similaires entre cartes physiques et cartes numériques, nous avons conservé l'outil caméra pour ramasser une carte numérique, même s'il n'a aucune utilité fonctionnelle.

[32] ramasse

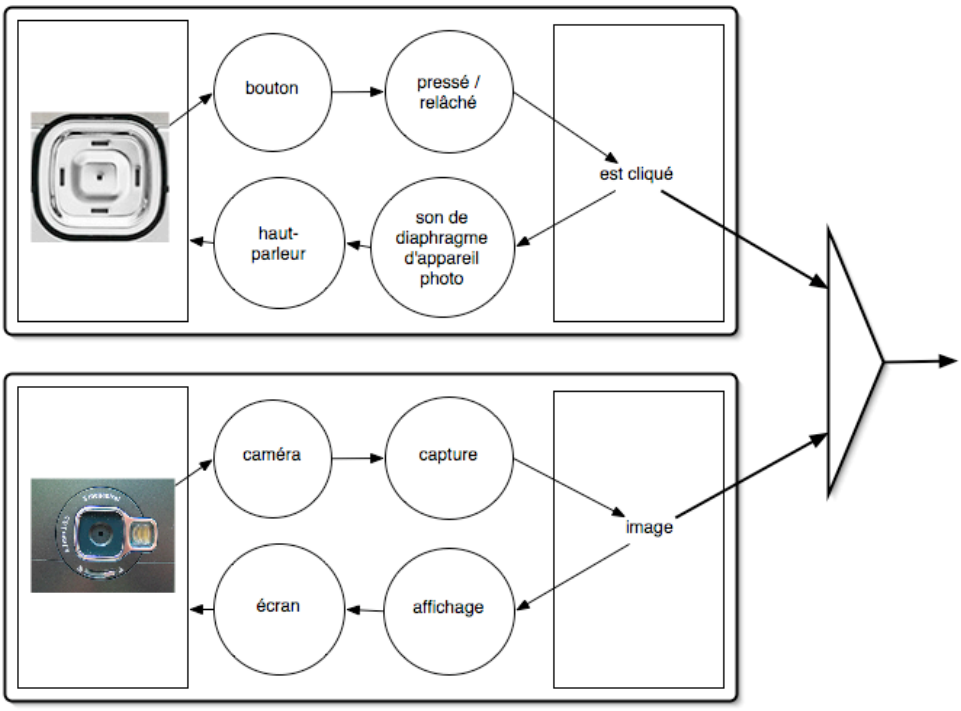


Figure 116 : Dispositifs d'interaction avec les cartes de Memory dans Snap2Play.

3.1.1.3.3. Synthèse de l'expérience de conception

Lors de cette expérience, nous avons utilisé le cadre fourni par le modèle d'interaction mixte de façon plus systématique pour explorer l'espace de conception. Notre objectif était, non seulement d'évaluer le pouvoir génératif du modèle, mais aussi de proposer une interaction similaire entre les cartes d'origine physique et d'origine numérique. Une évaluation a été conduite afin de savoir si notre intention de générer des techniques d'interaction similaires pour les deux types de cartes a effectivement été perçue comme telle par les utilisateurs. L'expérience est présentée dans [You et al., 2008]. Dans une enquête effectuée après la session d'expérience de jeu avec Snap2Play, les joueurs devaient évaluer leur expérience de ramassage des cartes. La majorité des joueurs a trouvé que ramasser une carte physique et une carte numérique était similaire (Figure 117, *overall interaction*[33]). De même, en détaillant les étapes de l'interaction, la majorité des joueurs a trouvé ces interactions similaires (Figure 117) :

- Trouver l'emplacement de la carte (Figure 106, écrans 1-2-3, et *finding the card location* Figure 117),
- Trouver l'orientation de la carte une fois l'emplacement trouvé (Figure 106, écran 4, et *finding the card position within the visual space*, Figure 117),
- Positionner le téléphone pour prendre la photo une fois l'orientation de la carte trouvée (Figure 106, écran 4 à droite, et *positioning the card for photo taking once card found*, Figure 117)
- Prendre en photo la carte (Figure 106, écran 5, et *taking photo of the card*, Figure 117).

[33] interaction générale

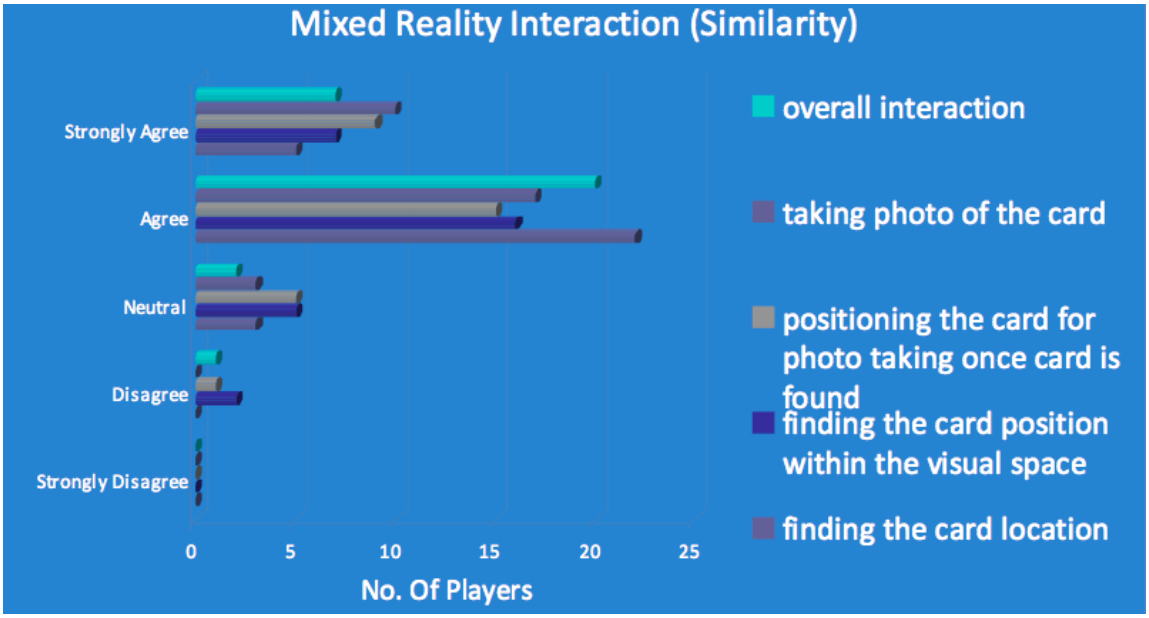


Figure 117 : Résultats de l'enquête qui a suivi les expériences de jeu avec Snap2Play.

3.1.2. Conception par un expert et un novice

3.1.2.1. ORBIS : un système pour visionner ses photos

Le modèle d'interaction mixte à notre disposition pour concevoir ORBIS était plus complet que pour les systèmes présentés précédemment. Ce système a été conçu en collaboration avec un designer. Il s'agit d'une première expérience ou au moins un des deux utilisateurs n'était pas expert dans le modèle.

ORBIS est un système visant à fournir de nouveaux moyens de profiter de ses données personnelles, comme les photos, la musique, les vidéos. La liste de photos est importée dans le système au préalable. Dans notre première version d'ORBIS, nous n'avons considéré que les photos. Les photos sont embarquées dans un objet en silicone (Figure 118), et affichées sur un petit écran. Quelle que soit l'orientation de l'objet dans le plan vertical, la photo est toujours redressée par le système grâce à des accéléromètres qui captent l'orientation de l'objet (Figure 119).

L'objet de la tâche de l'utilisateur est la liste de photos. Nous présentons sa conception dans un premier temps. ORBIS permet à l'utilisateur d'interagir avec cet objet : il permet notamment de démarrer ou arrêter l'affichage des photos, naviguer dans les photos, et mélanger les photos, grâce à des outils dont nous présentons la conception dans un second temps.

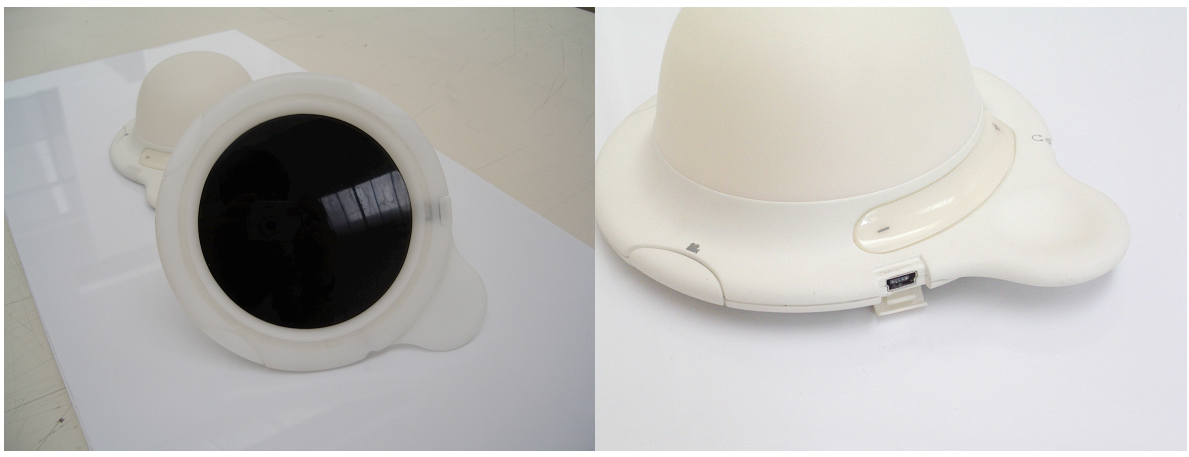


Figure 118 : Prototypes de haute fidélité en silicone (non interactif) du système ORBIS.

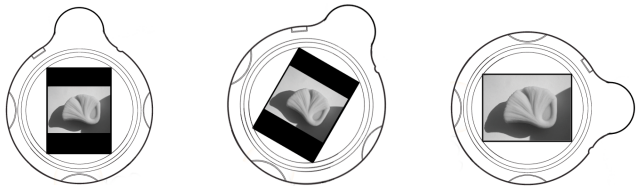


Figure 119 : Quelle que soit l'orientation de l'objet dans le plan vertical, la photo affichée est toujours dans le bon sens.

3.1.2.1.1. Conception de l'objet de la tâche : la liste de photos

Dans le cas de la conception d'ORBIS, nous avons observé que le modèle d'interaction mixte nous a permis d'amorcer la conception. Comme la liste de photos est un objet originellement numérique, nous avons commencé par les propriétés numériques de l'objet. La première propriété numérique identifiée est la liste de photos numériques : Image0, …, ImageN. Nous avons identifié d'autres propriétés numériques attachées à la liste de photos :

- L'ordre des photos, initialement 0, …, N ;
- Une propriété indiquant si les photos sont présentées ou non à l'utilisateur : « est présentée », initialement fausse;
- La photo courante, initialement 0.

La Figure 120 présente cette première étape de conception avec le modèle d'interaction mixte.

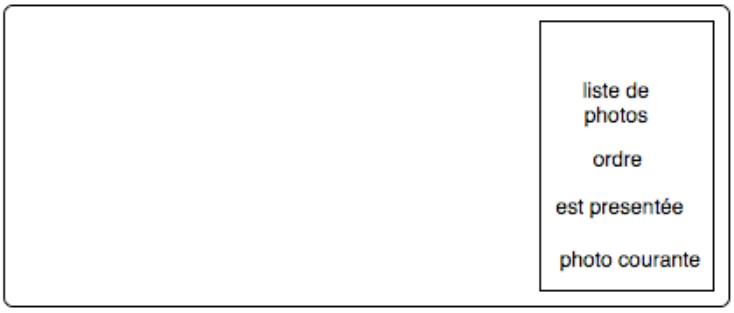


Figure 120 : Objet de la tâche d'ORBIS : première étape de conception.

En suivant les caractéristiques du modèle d'interaction mixte, nous voyons que ces propriétés peuvent être acquises et/ou matérialisées. Comme nous souhaitons ancrer l'objet dans le monde physique, nous l'avons augmenté d'une partie physique et nous avons matérialisé les propriétés numériques non acquises.

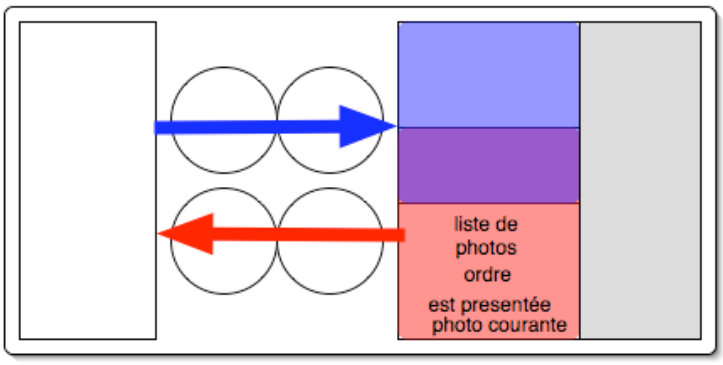


Figure 121 : Objet de la tâche d'ORBIS : les propriétés numériques identifiées dans la première étape de conception ne sont pas acquises, mais sont matérialisées.

Pour matérialiser les propriétés physiques, nous pouvons concevoir plusieurs modalités de liaison en sortie (Figure 122). Nous avons d'abord choisi les propriétés physiques générées par le dispositif

de liaison, parmi toutes les possibilités (son, etc.). Nous avons évidemment choisi une modalité visuelle pour matérialiser les photos.
Ensuite, nous avons considéré les dispositifs permettant de générer des propriétés physiques qui se voient, et nous avons imaginé plusieurs solutions : un projecteur, un mini-écran, ou des lunettes semi-transparentes (Figure 122). Nous avons choisi le dispositif le plus petit avec le maximum de résolution : le mini-écran.

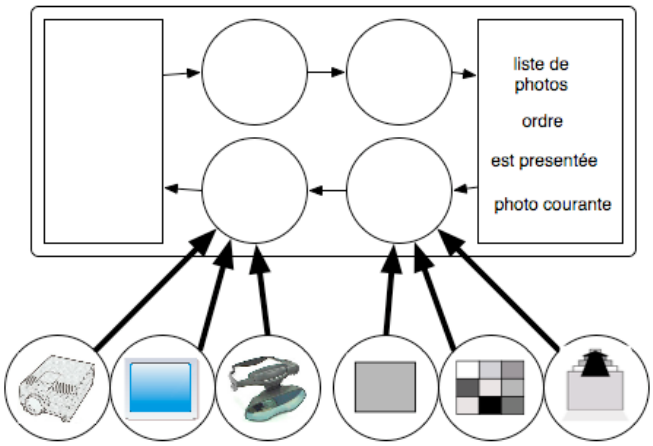


Figure 122 : Exploration des possibilités de conception pour la modalité de liaison en sortie.

Enfin, nous avons choisi un langage non linguistique, puisque nous cherchions à matérialiser directement des images. Nous avons choisi d'afficher les photos en deux dimensions et en couleur, de façon non déformée. Pour compenser la petite taille de l'écran, nous avons choisi d'avoir un langage local (section 2.1.1.2 du Chapitre 4), c'est-à-dire n'affichant pas la totalité des images, mais les affichant une par une, au lieu de toutes à la fois. Pour compenser cette caractéristique locale du langage, nous avons choisi un langage dynamique (section 2.1.1.2 du Chapitre 4) pour que l'utilisateur puisse tout de même voir toutes les photos. Toutes ces caractéristiques nous ont conduit à choisir le diaporama (Figure 122, langage de gauche). Nous avons envisagé d'autres alternatives de conception, par exemple un tableau photos (Figure 122, langage du milieu), qui n'est pas local et dynamique, mais global et statique. Nous aurions pu également choisir un langage qui affiche les photos sous forme d'une file en trois dimensions (Figure 122, langage de droite), qui est global, déformé, et dynamique.

Nous avons ensuite poursuivi la conception avec les propriétés physiques de l'objet. Les propriétés physiques peuvent être capturées ou générées par des modalités de liaison. Le choix des propriétés physiques ni capturées ni générées (le matériau ou la forme, Figure 123) a été guidé par l'esthétique et la portabilité. Nous retrouvons, à ce niveau physique, l'image générée par la modalité de liaison en sortie. Nous avons également considéré d'autres propriétés, comme l'orientation de l'image générée dans le plan vertical. L'objectif est que cette image soit toujours correctement orientée, quelle que soit l'orientation de la partie en silicone de l'objet (Figure 119). Pour ce faire, nous avons besoin de capter l'orientation de la partie en silicone de l'objet (Figure 123).

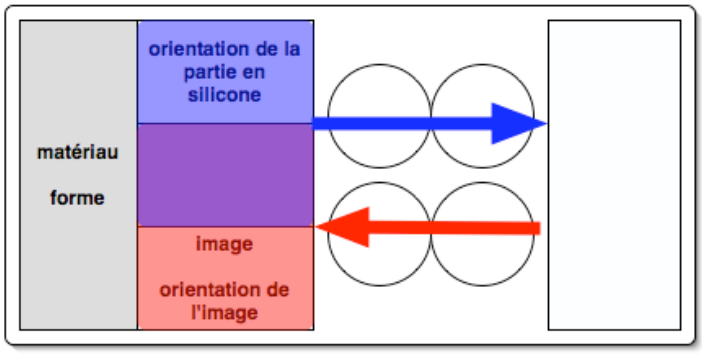


Figure 123 : Conception des propriétés physiques de l'objet de la tâche dans ORBIS.

L'orientation de la partie en silicone de l'objet devant être captée, nous devons donc concevoir une modalité de liaison en entrée, liant l'orientation de la partie physique l'objet de la tâche à l'orientation des photos numériques. Nous avons de nouveau plusieurs possibilités pour concevoir cette modalité de liaison en entrée. La Figure 124 présente les différentes solutions imaginées. Nous avons choisi de prototyper la modalité qui pouvait être embarquée dans l'objet : celle avec les accéléromètres.

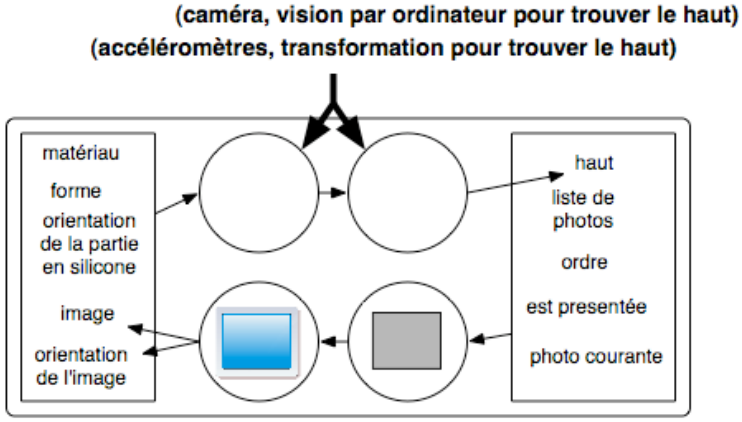


Figure 124 : Concevoir une modalité de liaison en entrée pour lier l'orientation de l'objet avec celle des photos.

Une fois cette modalité de liaison conçue, l'objet a alors une nouvelle propriété numérique : haut (Figure 124). Cette propriété numérique a quatre valeurs possibles (haut, bas, droite, gauche), selon le côté de la photo qui doit être présenté en haut. La Figure 125 présente la conception finale de l'objet de la tâche d'ORBIS (la liste de photos mixte) décrit avec le modèle d'interaction mixte.

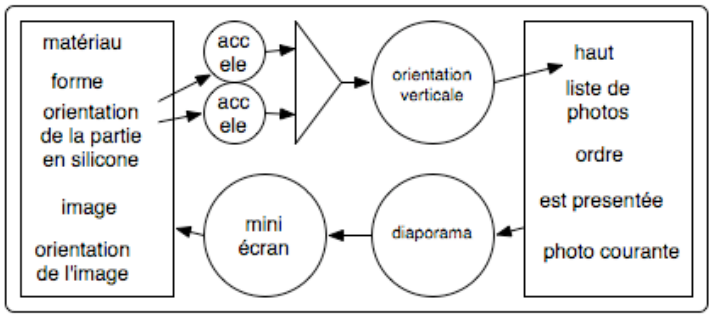


Figure 125 : Conception finale de l'objet de la tâche d'ORBIS : la liste de photos mixte.

Une fois cet objet conçu, nous nous sommes intéressés aux modalités d'interaction avec cet objet (1) pour démarrer ou arrêter la présentation de la liste, (2) pour naviguer dans les photos, et (3) pour mélanger les photos.

3.1.2.1.2. Conception des outils pour démarrer/arrêter la présentation, naviguer dans, et mélanger la liste de photos

Pour concevoir les modalités d'interaction avec l'objet de la tâche pour ces trois tâches, nous devons concevoir 3 outils. Pour ces trois outils, nous avons notamment exploré les métaphores possibles (section 3.2.5 du Chapitre 4), ainsi que les ports physiques et numériques (section 3.2.6 du Chapitre 4) et la composition spatiale entre les objets (section 3.2.4 du Chapitre 4). Nous présentons maintenant leur conception.

3.1.2.1.2.1. Concevoir un outil pour démarrer/arrêter la présentation

Pour concevoir l'outil pour démarrer/arrêter la présentation, nous avons considéré des solutions avec ou sans métaphore de nom (Tableau 17, à gauche). Pour cela, nous avons cherché comment les propriétés physiques peuvent refléter la tâche « démarrer/arrêter la présentation ».
Tout d'abord, les propriétés physiques peuvent ne pas rappeler la tâche « démarrer/arrêter la présentation » (Tableau 17, en haut à gauche). Dans ce cas, il n'y a pas de métaphore de nom car la forme de l'objet ne reflète pas la tâche. Ensuite, les propriétés physiques peuvent ressembler à un interrupteur électrique (Tableau 17, en bas à gauche). Elles définissent dans ce cas une métaphore de nom car un interrupteur électrique dans ORBIS est alors comme un interrupteur électrique dans le monde réel : il permet d'allumer ou d'éteindre. Pourtant, ce n'est pas forcément une métaphore de verbe, car rien n'est encore décidé pour la propriété numérique acquise de cet outil (qui correspond à l'action faite sur l'outil).

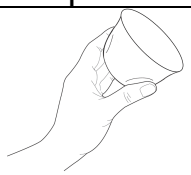
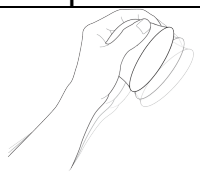
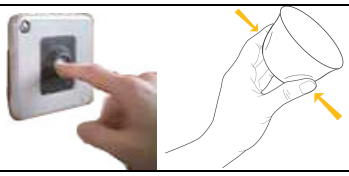
	Métaphore de nom	**Métaphore de verbe**
Absence de métaphore		
Présence de métaphore		

Tableau 17 : Explorer les alternatives de conception en fonction des métaphores possibles.

Nous avons ensuite considéré des solutions avec ou sans métaphore de verbe (Tableau 17, à droite). Pour cela, nous avons cherché comment les propriétés numériques pouvaient refléter la tâche « démarrer/arrêter la présentation ». Tout d'abord, les propriétés numériques peuvent ne pas rappeler la tâche « démarrer/arrêter la présentation » (Tableau 17, en haut à droite). Dans ce cas, la propriété numérique acquise « est secoué » ne rappellent pas la tâche. Ensuite, les propriétés numériques peuvent refléter la tâche « démarrer/arrêter la lecture » : dans le cas du Tableau 17 en bas à droite, la propriété numérique acquise « est pressé » reflète la tâche « démarrer/arrêter », car l'utilisateur appuie sur l'objet comme il appuie sur l'interrupteur pour démarrer ou arrêter.

Nous avons finalement choisi de ne pas proposer de métaphore de nom pour des raisons esthétiques, et de proposer une métaphore de verbe. La conception finale choisie est présentée Figure 126. L'utilisateur applique une pression sur les propriétés physiques de l'outil. Cette pression est captée par une modalité de liaison (capteur de pression, seuil) pour mettre à jour l'état (propriété numérique) « est pressé » de l'outil (Figure 126). Le modèle incite et guide alors pour considérer une réaction de l'outil : nous avons donc imaginé une réaction de l'outil avec une modalité de liaison en sortie (haut-parleur, bip).
Selon la valeur de la propriété numérique « est pressé », le langage d'interaction va appliquer la tâche élémentaire à la liste de photos : si « est pressé » a été modifié et est vrai, alors le langage d'interaction change l'état « est présentée » de la liste de photo. Là encore, le modèle incite et guide pour considérer une réponse de l'objet de la tâche vers l'outil : nous avons donc conçu un langage

d'interaction en sortie qui modifie la propriété numérique « est ok » de l'outil lorsque « est présentée » a changée.
Finalement, afin de faire participer cette réponse à la réaction de l'outil, nous avons conçu une modalité de liaison en sortie qui affiche une image avec un symbole de lecture ( ) ou d'arrêt ( ) sur un écran.

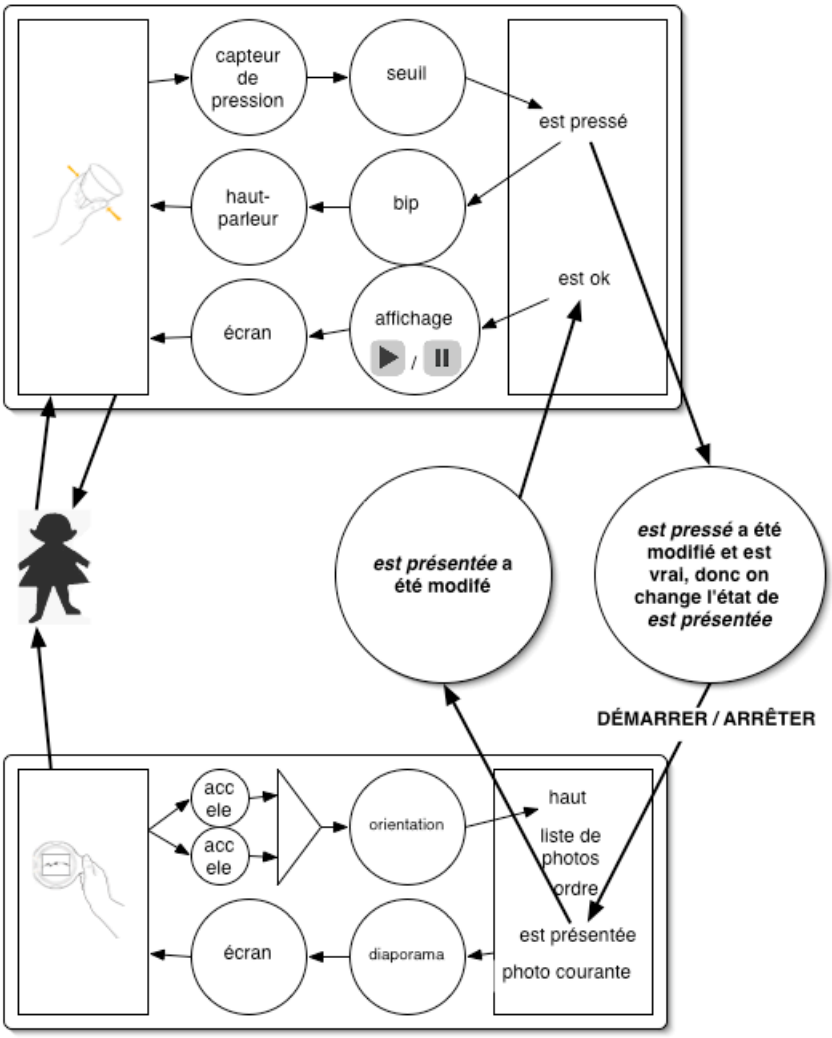


Figure 126 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « démarrer/arrêter la présentation ».

Enfin, nous avons considéré différentes compositions spatiales (section 2.2.4 du Chapitre 4) entre l'outil et l'objet de la tâche. Les objets adjacents et séparés sont présentés au Tableau 18. Nous avons adopté la solution adjacente (objets collés) afin que l'utilisateur puisse n'avoir qu'un objet dans les mains.

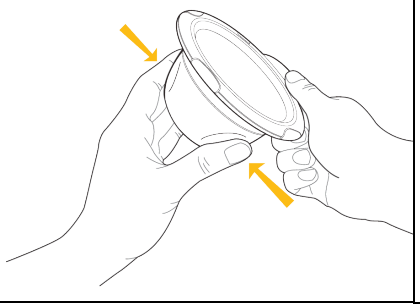
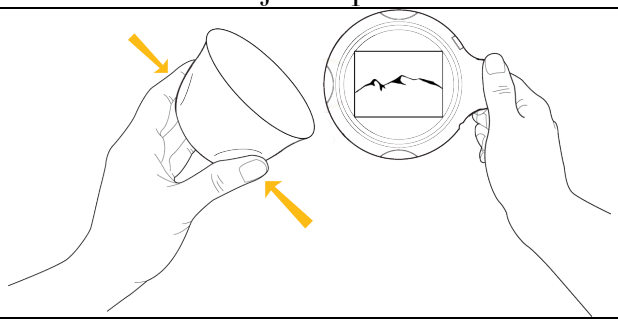
Objets adjacents	Objets séparés
	

Tableau 18: Compositions spatiales entre l'outil pour démarrer/arrêter la présentation et l'objet de la tâche

3.1.2.1.2.2. Concevoir un outil pour mélanger les photos

Pour l'outil qui sert à mélanger les photos, nous avons exploré une métaphore de verbe. Pour construire une métaphore de verbe, nous étudions comment les propriétés numériques acquises peuvent refléter la tâche « mélanger ». Nous avons pensé à la propriété numérique « est secoué ». Il y aurait alors une analogie entre mélanger les photos de la liste pour ORBIS et mélanger les dés dans un verre pour le jeu. On obtient alors une métaphore de verbe : secouer l'outil est comme secouer les dés, cela permet de mélanger.

La Figure 127 présente la technique d'interaction que nous avons conçue. L'utilisateur secoue l'outil. Les secousses sont captées par une modalité de liaison en entrée (accéléromètre, seuil et répétition) : lorsque l'utilisateur secoue l'outil plusieurs fois, la propriété numérique « est secoué » est mise à jour. Le modèle d'interaction mixte nous incite et nous guide à concevoir une modalité de liaison en sortie pour que l'utilisateur puisse percevoir une réaction de l'outil. Nous avons donc imaginé une réaction de l'outil avec une modalité de liaison en sortie (haut-parleur, bip).

Lorsque la propriété numérique « est secoué » de l'outil est mise à jour, alors le langage d'interaction applique la tâche élémentaire « mélanger » à la liste de photos. Là encore, le modèle d'interaction mixte met en évidence la possibilité de concevoir une réponse : lorsque l'ordre des photos a changé, le langage d'interaction en sortie met à jour la propriété numérique « est ok » de l'outil. Pour que la réponse de l'outil soit perceptible par l'utilisateur, nous avons conçu une modalité de liaison pour l'outil qui permet d'apprécier grâce à une image affichée sur un écran que la tâche élémentaire a bien été appliquée. De plus, le changement d'état de l'objet de la tâche est aussi perceptible via sa modalité de liaison en sortie : le diaporama montre que l'ordre des photos a été modifié.

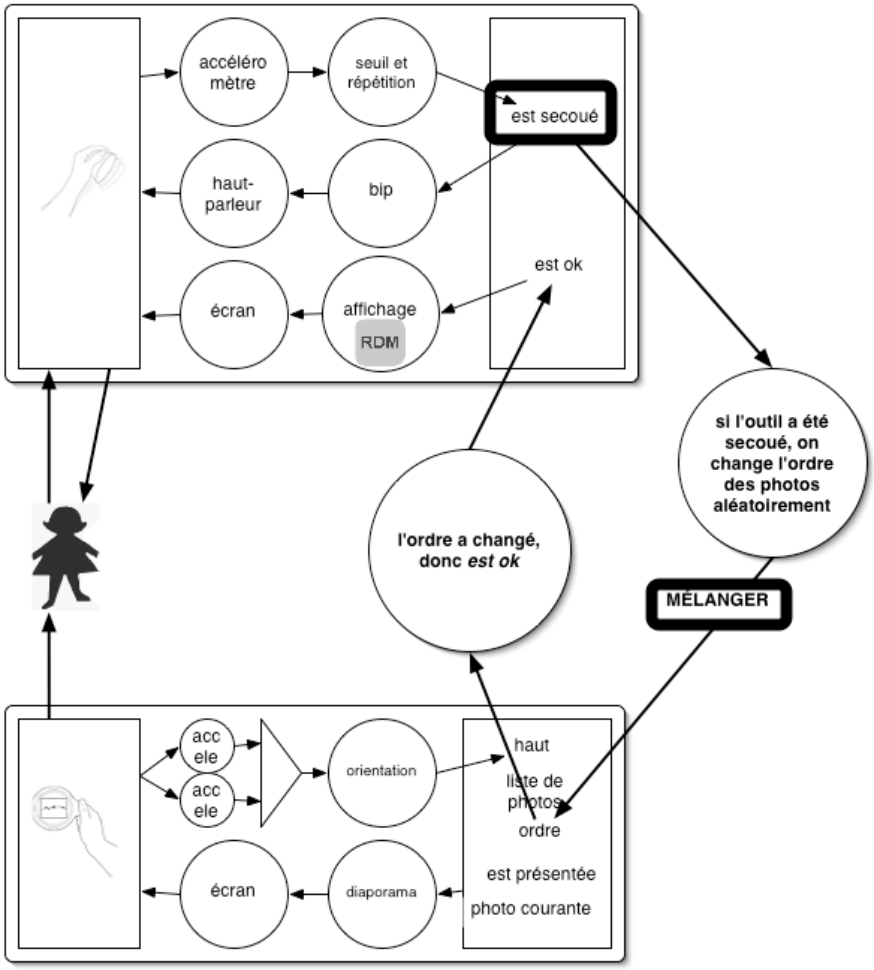


Figure 127 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « mélanger les photos ».

Enfin, nous avons considéré de la même façon que pour l'objet précédent deux compositions spatiales entre l'outil et l'objet de la tâche. Ces deux solutions envisagées sont présentées au Tableau 19.

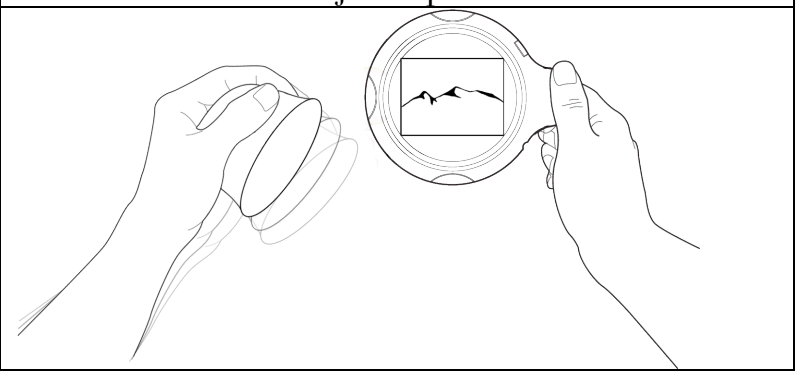
Objets adjacents	Objets séparés
	

Tableau 19: Compositions spatiales entre l'outil pour mélanger les photos et l'objet de la tâche.

3.1.2.1.2.3. Concevoir un outil pour naviguer dans les photos

Nous détaillons maintenant comment nous avons exploré les ports physiques et numériques pendant la conception de l'outil pour naviguer dans les photos. Les ports d'entrée ou de sortie sont des caractéristiques des propriétés physiques et numériques d'un objet mixte (section 2.2.6 du Chapitre 4). Ce sont des propriétés qui permettent à l'objet de « communiquer » avec l'extérieur.

Pour cet outil, nous avons conçu l'interaction présentée à la Figure 128. L'utilisateur agit sur la position angulaire de l'outil. Cette action entraîne la modification de la propriété numérique « angle ». Il n'y a pas de réaction directe et la propriété numérique « angle » n'est pas perceptible.
Le langage d'interaction traduit la nouvelle valeur de la propriété numérique « angle » pour mettre à jour la photo courante de l'objet de la tâche. L'utilisateur perçoit la modification de l'état interne de l'objet mixte via le retour d'information : la photo présentée a changé. L'utilisateur perçoit également le changement d'état de l'objet de la tâche via la réponse vers l'outil : à chaque fois que la photo courante change, la propriété numérique « cran » est modifiée dans l'outil, via le langage d'interaction en sortie. L'utilisateur perçoit les « crans » grâce à deux réactions de l'outil : le servomoteur qui met sa position à jour en suivant la valeur de « cran », et aussi grâce à un son entendu dans le haut-parleur. L'outil a donc une liaison en sortie multimodale pour rendre perceptible la réponse de l'objet de la tâche vers l'outil.

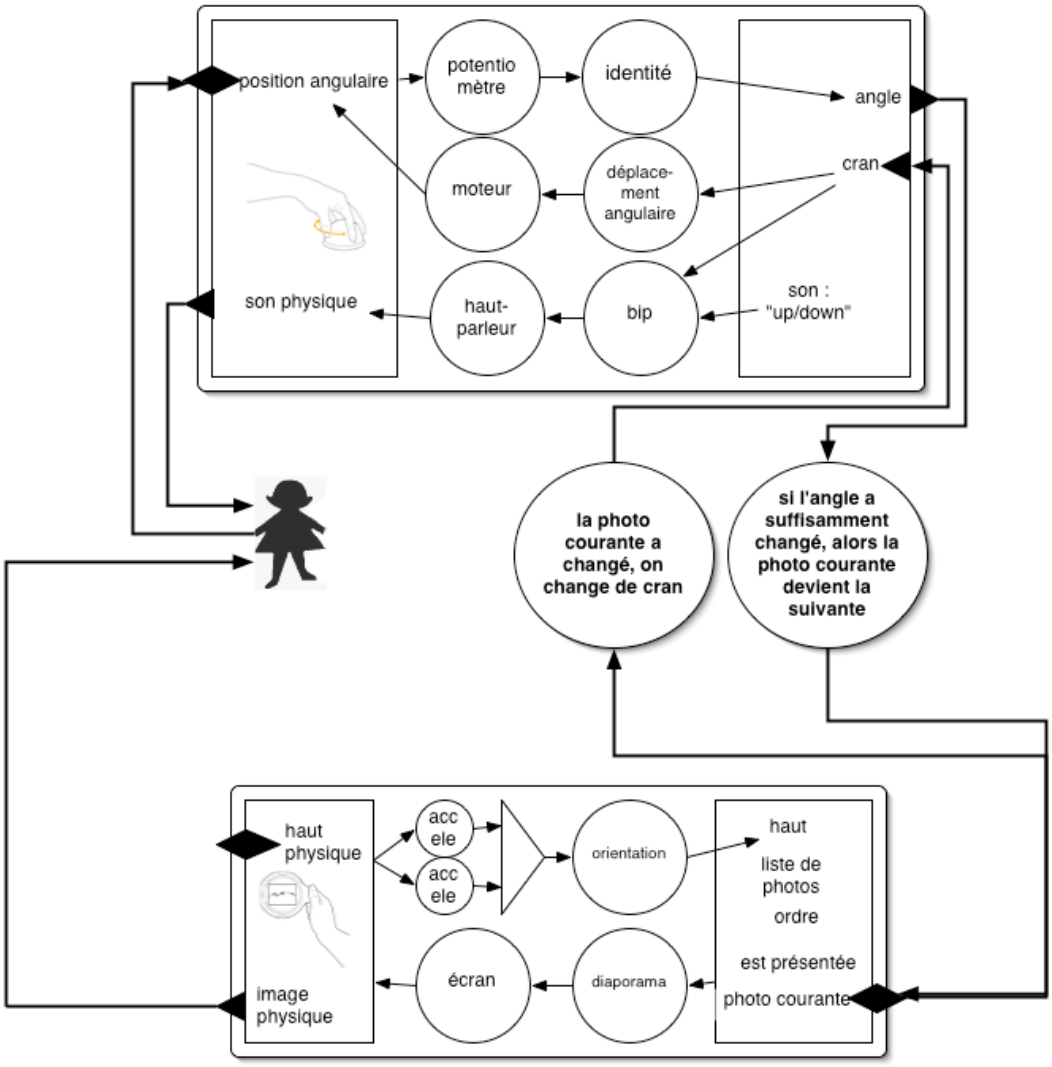


Figure 128 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « naviguer dans les photos ».

Pour explorer les ports d'entrée/sortie physique et numérique, nous considérons les propriétés physiques et numériques des objets. Dans le schéma de la Figure 128, les ports sont représentés par des triangles noirs.
Les ports d'entrée sont les propriétés physiques ou numériques, qui peuvent être modifiées par le monde physique ou par les langages d'interaction. Pour l'outil de la Figure 128, nous avons d'abord cherché quelle était l'affordance de l'objet physique, et comment il pouvait être facilement tourné. Nous avons donc décidé de fixer sa position angulaire comme port d'entrée physique. De plus, nous

nous sommes assurés que ce port ne soit pas contraint dans les différents contextes d'utilisation de l'outil, présentés Figure 129. Tenu dans la main (Figure 129, gauche), l'outil peut toujours être tourné. Posé sur la table(Figure 129, droite), l'outil peut toujours être tourné.

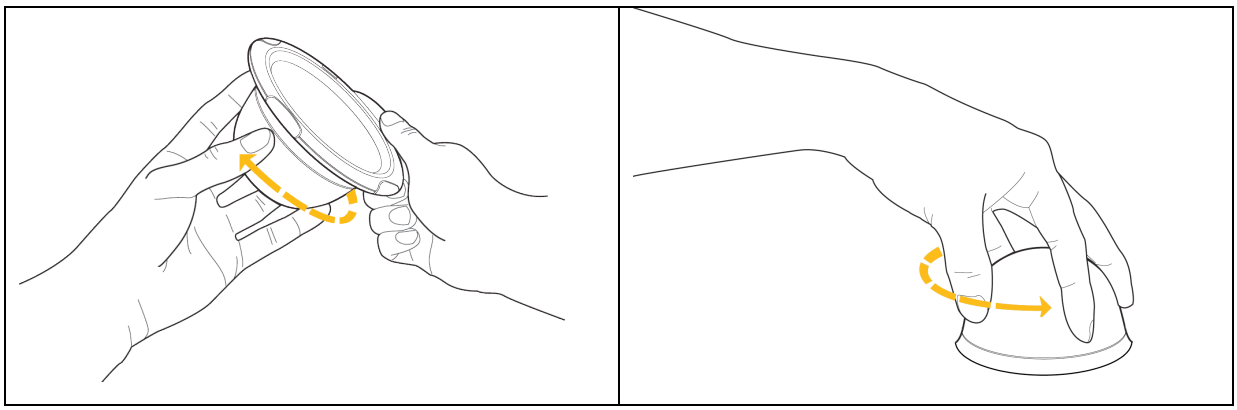


Figure 129 : Deux contextes différents pour l'utilisation d'ORBIS : tenu dans la main (gauche) ou posé sur une table (droite).

Ensuite, toujours pour la conception de cet outil, nous avons considéré le besoin d'avoir un port d'entrée numérique (*cran*) pour permettre à la réponse de l'objet de la tâche de participer à la réaction de l'outil. Ainsi l'utilisateur pourra mieux contrôler son action sur l'outil.
En considérant les ports de sortie de l'outil, nous avons remarqué que l'outil pour pouvoir agir sur l'objet de la tâche, devait avoir un port de sortie numérique (*angle*). De plus, nous remarquons que l'angle physique est un port physique à la fois d'entrée et de sortie. Ceci permet à l'utilisateur d'avoir une réaction précise grâce à la propriété physique sur laquelle il agit.

Enfin, nous avons considéré de la même façon que pour les objets précédents, deux compositions spatiales entre l'outil et l'objet de la tâche. Ces deux solutions envisagées sont présentées au Tableau 20 : soit l'outil et l'objet de la tâche sont adjacents (Tableau 20, à gauche), soit l'outil et l'objet de la tâche sont séparés. Nous avons misé sur une solution qui réduisait la discontinuité spatiale, et nous avons donc retenu les objets adjacents.

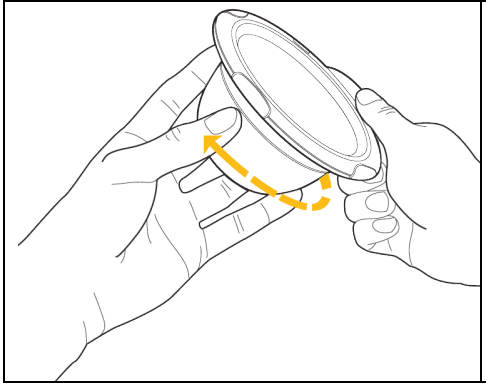
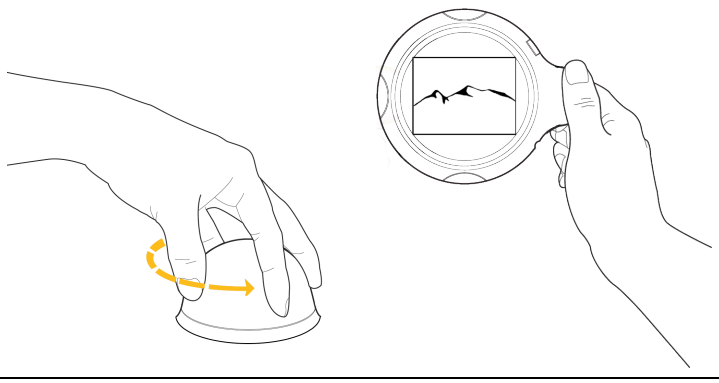
Objets adjacents	Objets séparés
	

Tableau 20 : Compositions spatiales entre l'outil pour naviguer et l'objet de la tâche.

3.1.2.1.3. Synthèse sur le pouvoir génératif après l'expérience d'ORBIS

Durant cette expérience avec ORBIS, nous avons utilisé une version plus avancée des caractéristiques du modèle d'interaction mixte. Par conséquent, nous avons pu explorer plus d'alternatives de conception. Le modèle nous a aussi été utile pour démarrer les discussions sur la conception et concevoir les premières solutions.

Néanmoins, il s'agissait là encore d'une utilisation du modèle en tandem entre un expert du modèle et un designer (Arts Appliqués). Cette expérience nous a donc seulement permis d'évaluer le pouvoir génératif du modèle si le modèle est complètement acquis et exploité par un expert. Nous avons néanmoins présenté le modèle au designer, qui l'a compris en partie : en effet nous avons remarqué qu'il a seulement considéré le niveau intrinsèque d'un objet mixte dans sa présentation finale du projet ORBIS. De plus, nous ne l'avons pas vu manipuler le modèle explicitement lors de la conception, bien que des questions de conception issues du modèle d'interaction mixte ont été considérées.

À cette étape de l'évaluation et de la construction simultanées du modèle d'interaction mixte, mettre le modèle dans les mains d'un designer seul nous a paru prématuré. Pour étudier plus en avant l'usage explicite du modèle par un designer non expert, nous avons mené une nouvelle expérience, avec un autre designer que nous présentons dans le paragraphe suivant.

3.1.2.2. Roam : un système d'enregistrement de souvenirs sonores

Pour concevoir *Roam*[34], nous avions à notre disposition l'ensemble des caractéristiques présentées au chapitre précédent. Cet exemple nous permet donc d'illustrer plus précisément du modèle d'interaction mixte.
Roam est un système que nous avons conçu en collaboration avec un designer (arts appliqués) différent de celui qui a participé à l'expérience précédente. Notre intention était de fournir à l'utilisateur un système mobile d'enregistrement sonore et photographique alternatif, qui ne le distrait pas de ce qui l'entoure, au contraire des appareils photos traditionnels. Avec un appareil photo, l'utilisateur concentre son attention sur l'outil plutôt que sur le monde qui l'entoure. Au contraire, Roam vise à rester au second plan de l'attention de l'utilisateur et ne pas le distraire de ce qui l'intéresse.
La Figure 130 et la Figure 131 présentent des illustrations du prototype de l'outil fourni par le système Roam pour enregistrer des souvenirs sonores. L'outil trouve sa place dans la main de l'utilisateur sans le gêner (Figure 130). L'utilisateur peut plier avec le pouce une partie de l'outil pour commencer l'enregistrement, et la relâcher pour le terminer. Une diode électroluminescente jaune lui permet de savoir si la partie est assez pliée ou relâchée, pour commencer ou terminer l'enregistrement. Une diode électroluminescente verte indique si l'enregistrement a bien lieu. L'écoute de l'enregistrement n'est pas prévue dans Roam mais peut-être faite avec un ordinateur, après l'utilisation de Roam. L'outil mixte focalise uniquement sur la capture.

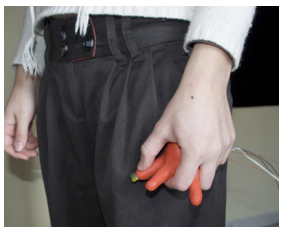


Figure 130 : Le prototype final de l'outil mixte dans Roam (Figure 131) dans la main de l'utilisateur.

[34] vadrouiller



Figure 131 : Le prototype final de l'outil mixte dans Roam, avec capteur de flexion et diodes électroluminescentes vertes et jaunes.

Pour la conception de l'outil mixte de capture dans Roam, nous présenterons d'abord comment nous avons exploré l'espace de conception pour sa partie extrinsèque, puis sa partie intrinsèque.

3.1.2.2.1. Conception extrinsèque

L'outil présenté aux Figure 130 et Figure 131 doit être utilisé dans un contexte mobile, et vise à permettre à l'utilisateur d'enregistrer des souvenirs sonores sans être remarqué ou distrait de ce qui l'intéresse. Son rôle (outil) est évident, mais d'autres caractéristiques extrinsèques restent à explorer. Nous présentons comment l'espace de conception du modèle d'interaction mixte nous a permis d'explorer des métaphores de nom et d'étudier les ports physiques et numériques.

L'exploration des métaphores de nom est résumée au Tableau 21. La tâche est « enregistrer le son ». Nous avons exploré les métaphores de nom beaucoup plus en détail que dans l'expérience précédente avec ORBIS. Une métaphore de nom peut faire référence à la commande (« enregistrer ») comme dans la première ligne du Tableau 21. Elle peut aussi faire référence au paramètre de la commande comme dans la dernière ligne du Tableau 21. Nous avons également identifié un continuum entre les métaphores faisant référence au monde « réel » et celles qui font référence aux pratiques numériques. Pour l'outil de Roam, ces caractéristiques du modèle nous ont permis de générer six solutions présentées dans le Tableau 21. Nous avons finalement choisi la métaphore de la pieuvre puisqu'elle trouvent mieux sa place dans la main de l'utilisateur.

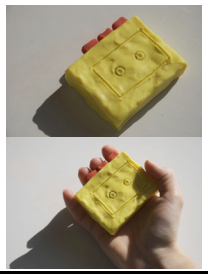
	Pratique « naturelle »	**↔**	**Pratique numérique**
Commande (enregistrer)	Pieuvre (enregistrer comme manger/attraper: la pieuvre mange grâce à son bec et attrape les éléments de son environnement avec ses tentacules) 	Manivelle des anciennes caméras (enregistrer comme dérouler le film) 	Point rouge (symbole de l'enregistrement dans les systèmes comme les caméscopes) 
Paramètre (son)	Oreille 	Pavillon d'un phonographe 	Magnétophone 

Tableau 21 : Exploration de l'espace de conception pour l'outil d'enregistrement sonore de Roam, selon les différentes possibilités de métaphores de nom. La tâche est « enregistrer le son ».

Pendant la conception, nous avons également étudié explicitement les ports, physiques et numériques, de cet outil.

Tout d'abord, nous avons eu besoin d'un port de sortie numérique : « isOn ». Nous avons également identifié la possibilité d'avoir un port d'entrée numérique grâce à notre espace de conception. Cette possibilité nous a inspirés l'idée de fournir une réponse de l'objet de la tâche (l'enregistrement) vers l'outil : l'outil contient alors une propriété numérique qui transmet le succès de la commande « enregistrer ». Si cette propriété numérique est matérialisée ensuite, alors l'outil fournit une meilleur observabilité [Abowd et al., 1992]. Nous avons donc choisi d'avoir un port d'entrée numérique, appelé « isOk ».

En considérant les ports physiques, nous avons d'abord exploré les possibilités de ports de sortie physiques. Le modèle nous a inspiré plusieurs alternatives, comme le son (Figure 132 à gauche) ou la lumière (Figure 132 au centre et à droite). La Figure 132 présente ces ports avec des dispositifs particuliers, mais les ports (extrinsèques) sont conçus indépendamment des dispositifs (intrinsèques). Nous pourrions par exemple imaginer d'autres dispositifs lors de la conception intrinsèque pour avoir la lumière comme port de sortie : une lampe, un laser, un écran, etc.

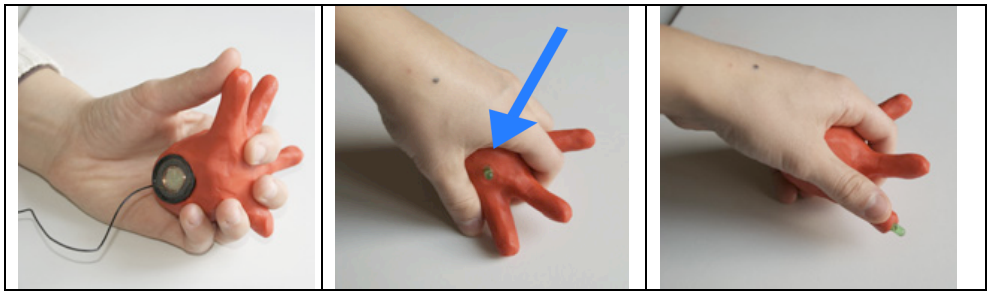


Figure 132: Alternatives de conception pour les ports de sorties physiques : le son avec l'exemple du haut-parleur (à gauche), la lumière (ici avec l'exemple des LEDs (au centre et à droite). En considérant les ports de sortie, physiques, nous constatons que selon la façon dont l'outil est pris en main par l'utilisateur, la lumière peut être cachée par sa main (au centre) ou non (à droite).

Nous avons choisi entre les alternatives de ports de sortie en considérant le contexte d'utilisation : cet outil mobile peut être pris en main de telle façon que la lumière de la Figure 132 au centre (au bout de la flèche) peut être cachée et ne pas être perceptible. De plus, le son peut être trop gênant, car les autres peuvent l'entendre. Nous avons donc choisi de la lumière placée comme à la Figure 132 à droite.

Nous avons présenté comment les caractéristiques extrinsèques associées au modèle d'interaction mixte permettent d'explorer l'espace de conception et d'envisager des solutions. L'exploration des caractéristiques extrinsèques nous a amené à étudier les caractéristiques intrinsèques. Nous avons identifié les ports de l'outil, nous étudions maintenant comment concevoir ces propriétés physiques et numériques au niveau intrinsèque.

3.1.2.2.2. Conception intrinsèque

La Figure 133 rappelle les ports d'entrée/sortie physiques et numériques de l'outil mixte conçus au niveau extrinsèque. Nous avons choisi « isOn » comme port numérique de sortie et « isOk » comme port numérique d'entrée, ainsi que des lumières au bout des tentacules comme ports physiques de sorties.

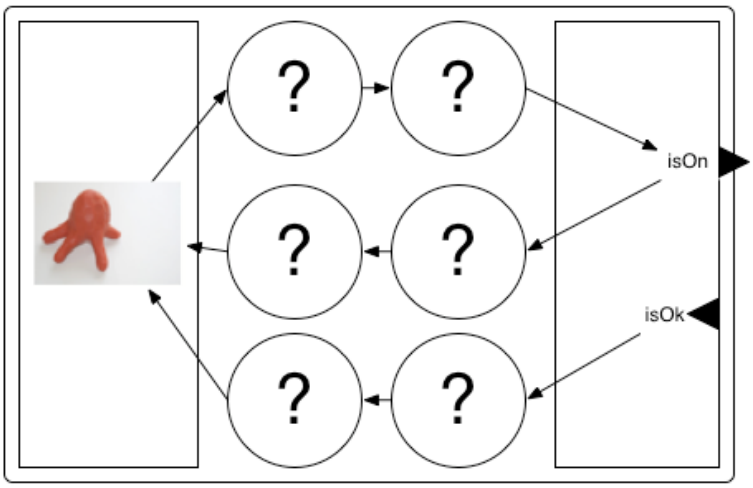


Figure 133 : Point de départ de conception intrinsèque de l'outil d'enregistrement de Roam.

En nous reposant sur les ports physiques et numériques étudiés, nous avons choisi comme solution de conception finale des modalités de liaison en sortie deux diodes vertes et jaunes associées à un langage non linguistique, global, précis, non déformant, et arbitraire :

- La diode jaune est allumée tant que « isOn » est vrai, et éteinte tant que « isOn » est faux.
- La diode verte est allumée tant que « isOk » est vrai, et éteinte tant que « isOk » est faux.

Nous considérons maintenant les propriétés numériques. Nous avons déjà identifié deux propriétés numériques : le port d'entrée « isOk » et le port de sortie « isOn » (Figure 133). Ces propriétés

numériques peuvent être à rebond, c'est-à-dire retrouver leur valeur initiale à partir d'un certain laps de temps. Si « isOn » est à rebond, alors la diode clignotera une fois. Au contraire si « isOn » n'est pas à rebond, alors la diode jaune sera allumée tant que le tentacule est suffisamment plié. Nous avons choisi d'avoir « isOn » sans rebond. Au contraire, nous avons choisi d'avoir « isOk » à rebond : l'utilisateur doit alors regarder la diode verte quand il plie le tentacule pour savoir si l'enregistrement a bien commencé.

Comme pour ORBIS, nous avons également considéré la composition spatiale des propriétés physiques. Selon l'espace de conception engendré par le modèle d'interaction mixte, nous pouvons envisager plusieurs solutions pour la composition des propriétés physiques de l'outil : elles peuvent être adjacentes (Figure 130 et Figure 131) ou séparées, manipulées par les deux mains (Figure 134).

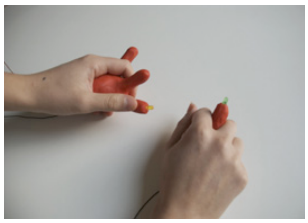


Figure 134 : Autre prototype de l'outil pour Roam, avec des propriétés physiques spatialement séparées.

3.1.2.2.3. Synthèse sur le pouvoir génératif dans l'expérience de conception de Roam

L'utilisation du pouvoir génératif du modèle d'interaction mixte nous a aidé dans cette expérience de conception avec un autre designer pour générer des alternatives de conception. Le modèle d'interaction mixte encourage à explorer l'espace des possibilités en identifiant des alternatives de conception. Nous avons choisi de souligner l'impact des caractéristiques identifiées dans la dernière version du modèle pour l'exploration de l'espace de conception. Néanmoins d'autres éléments du modèle ont été utilisés pour concevoir l'ensemble du système Roam comme l'affordance et les propriétés physiques captées (section 2.1.2 du Chapitre 4).
Cette expérience a confirmé la précédente, car nous n'avons pas vu non plus le designer manipuler le modèle explicitement lors de la conception, bien que nous ayons considéré ensemble des questions de conception guidées par le modèle d'interaction mixte. Afin de mieux comprendre comment un utilisateur non expert du modèle peut manipuler le modèle, nous avons demandé à un designer seul de l'utiliser et nous lui avons demandé de nous expliquer son expérience. Nous présentons donc maintenant la première expérience de conception avec un utilisateur non-expert du modèle d'interaction mixte.

3.1.3. Conception par un novice du modèle d'interaction mixte

3.1.3.1. Yubi : un système d'objets physiques augmentés pour accéder à des données numériques

Yubi est un système de réalité mixte conçu par un designer. Le système permettant à l'utilisateur de créer ses propres objets mixtes. Il peut associer des photos, de la musique, des vidéos aux objets de la vie courante, comme le coquillage rapporté de vacances de la Figure 135. Yubi fournit des marqueurs RFID sous forme d'autocollants pour équiper les objets (Figure 135). Pour utiliser ensuite ces objets mixtes, l'utilisateur les pose sur une surface de lecture interactive (lecteur RFID), et retrouve sur l'écran la partie numérique de l'objet. La Figure 136 présente l'interface graphique de Yubi : sur cet interface, on voit que le lecteur de marqueurs RFID a détecté un objet (« yubiz

détecté ») dont le code RFID est « dope687 ». On voit sur la Figure 136 qu'à cet objet mixte sont associés 17 éléments numériques. En cliquant sur la flèche (en bas à droite), on peut voir la liste de ces éléments et éventuellement les supprimer. Pour ajouter des éléments, il suffit de les glisser-déposer avec la souris dans la zone ronde et bleue centrale.



Figure 135 : Marqueurs RFID fournis du système Yubi posés par un utilisateur sur un objet personnel, souvenir de vacances.



Figure 136 : Interface graphique pour accéder aux données associées à un objet : nous observons que 17 éléments numériques sont associés à cet objet mixte. L'accès à liste de ces éléments se fait en cliquant sur la flèche (en bas à droite). Pour ajouter des éléments, il suffit de les glisser-déposer dans la zone ronde centrale.

Pour décrire l'interaction avec les objets augmentés de Yubi, nous considérons que l'objet mixte est formé d'un objet physique et de son marqueur RFID du côté physique (Figure 137), et de données numériques (photo, musique, vidéo) du côté numérique (Figure 137). L'utilisateur peut ensuite interagir avec cet objet via un outil (l'interface graphique de la Figure 136) activé par la souris. La Figure 137 présente la description de cet objet avec le modèle d'interaction mixte.

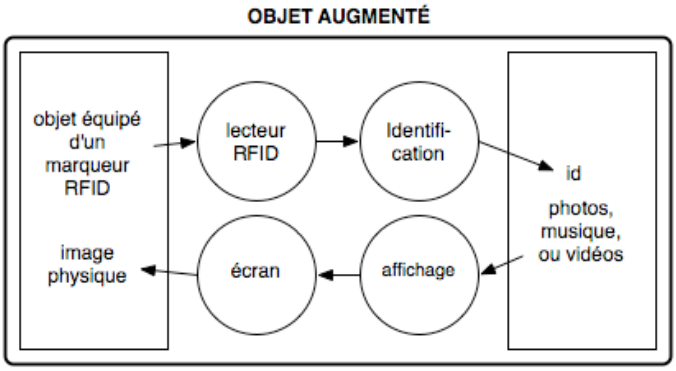


Figure 137: Description d'un objet augmenté avec Yubi selon le modèle d'interaction mixte.

Il convient de noter ici que d'autres modélisations sont possibles, et que le modèle laisse la liberté au concepteur de choisir celle qui lui semble pertinente. Par exemple, l'objet augmenté pourrait être considéré comme un outil (avec son lecteur) et non un objet de la tâche. L'objet de la tâche est alors purement numérique et l'utilisateur manipule un objet physique pour activer un contenu numérique, comme les Phicons. L'utilisateur associe dans ce cas un contenu numérique à un objet physique, mais pour s'en servir d'outil afin d'accéder à ce contenu sur l'écran de l'ordinateur ou de la télévision.

Pour que le designer utilise le modèle d'interaction mixte, nous lui avons fourni un support académique de présentation du modèle (un article de recherche), et aucune aide. La Figure 138 présente la description de l'objet augmenté avec Yubi qu'il nous a fourni à l'issue de son utilisation du modèle d'interaction mixte. Nous avons ensuite recueilli ses remarques sur l'utilisation du modèle.

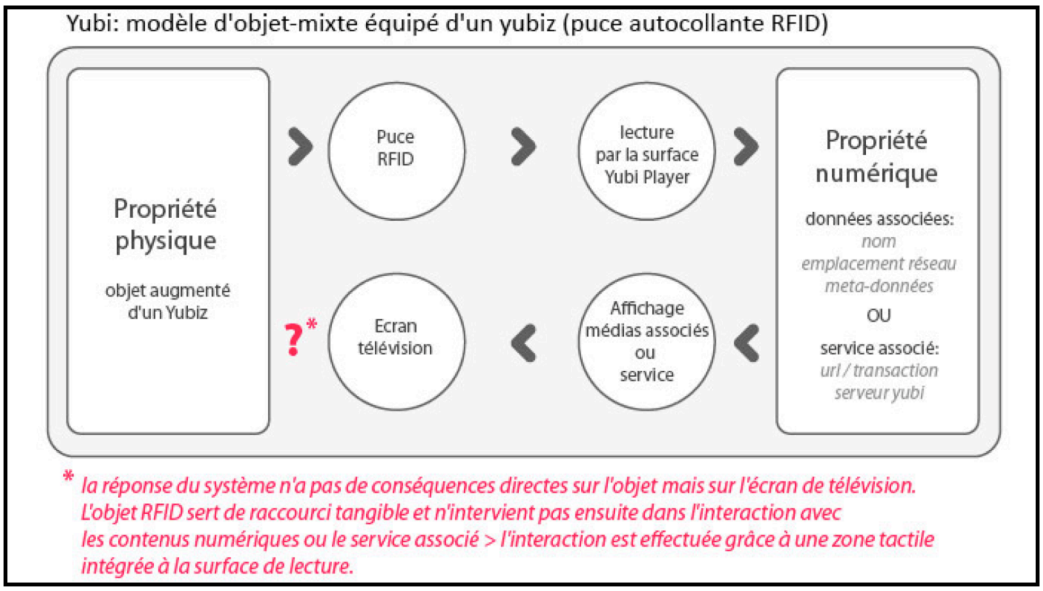


Figure 138 : Document produit par le designer du système Yubi : Description d'un objet mixte et entouré d'annotations.

Cette utilisation informelle du modèle d'interaction mixte par un non-spécialiste, non-informaticien, nous permet d'avoir un premier retour sur son utilisation possible. D'après son schéma (Figure 138) et les commentaires que nous avons recueillis, nous avons remarqué qu'il a pu utiliser le modèle, mais que des améliorations sont encore à apporter.
L'idée de modalité de liaison semble avoir plutôt bien été utilisée par le designer, puisque son schéma (Figure 138) est similaire au nôtre (Figure 137), et ses commentaires montrent qu'il a bien considéré la même modalité de liaison que nous : « je me suis intéressé au principe de "linking modality" décrit dans votre modèle, et qui permet dans l'interaction avec l'objet de faire lien entre ses propriétés physiques à ses propriétés numériques. Il s'avère que dans mon cas cette modalité semble unique, elle ne repose que sur le rapprochement de l'objet-mixte avec son lecteur », « Je me demandais donc si l'unique modalité d'entrée du système mixte de mon projet n'était pas la simple paire (puce RFID / dépot de l'objet sur le lecteur), qui permet au système d'acquérir les données numériques associées à l'objet (selon votre définition du couple "Device / Interaction Language"). La modalité de sortie serait, elle: Télévision (écran, son) / affichage des différents médias ou du service associé) ».
Au vue de son schéma (Figure 138) et du commentaire suivant : « Une fois l'objet activé, dans un second temps, l'utilisateur interagit avec les médias ou le service associé à son objet-mixte. Comme je me place dans une perspective de faisabilité à court terme, j'ai préféré opter pour une zone de défilement tactile intégrée à la surface de lecture afin d'interagir avec les éléments affichés sur la télévision de l'utilisateur », il semble avoir compris la différence entre modalité de liaison et modalité d'interaction.

Pourtant, il convient d'être prudent. En effet, l'expression « je me demandais si » et d'autres remarques, comme « est-ce réellement un objet-mixte ou bien un objet-augmenté ? », semble nous indiquer que le designer ne semble pas confiant dans son interprétation du modèle..
De plus, il semble ne pas comprendre qu'on puisse interagir avec l'objet mixte via ses propriétés numériques : « Il me semble pour autant que cette modalité supplémentaire d'interaction ne rentre pas dans le cadre du système mixte, puisque l'utilisateur ne manipule plus l'objet-mixte en question, mais plutôt son occurrence "digitale" à l'écran ». Ce dernier commentaire nous indique aussi que modéliser un objet mixte ayant des propriétés physiques spatialement distantes n'est pas évident. D'autre part il souligne la distinction franche que fait le designer entre système de réalité mixte et système interactif graphique. De son point de vue, dès lors que l'on revient a une interaction sur l'écran, le modèle ne s'applique plus.

À partir des résultats de cette première expérience d'utilisation du modèle d'interaction mixte, nous avons conclu que la compréhensibilité du modèle d'interaction mixte serait une condition nécessaire à son utilisation. De plus, même s'il est impossible d'en être sûr, le designer ne semble pas avoir utilisé le modèle pour explorer l'espace de conception.
L'ensemble des expérimentations menées, que ce soit avec un expert ou non du modèle, a permis d'observer comment le modèle peut servir de support à la conception, en particulier en incitant à explorer l'espace des possibilités. Toutes ces expérimentations sont non contrôlées, et pour aller plus loin dans l'évaluation du pouvoir génératif du modèle d'interaction mixte, nous avons effectué des évaluations avec protocole que nous présentons maintenant.

3.2. Évaluation avec protocole auprès des concepteurs

Au fil des évaluations non contrôlées présentées, nous avons noté qu'un enjeu important pour le modèle d'interaction mixte résidait dans son utilisation par des non experts. Les expériences de conception en situation réelle ne nous ont pas appris suffisamment pour être capable de comprendre comment le modèle a été utilisé par les novices. Nous avons donc préparé deux protocoles d'études du modèle d'interaction mixte. La première évaluation est un « groupe focalisé » sur la conception d'un système pendant deux heures. À la suite des expériences avec Yubi et avec ce groupe focalisé, nous avons touché du doigt un problème connu et confirmé par nos expériences de compréhension du modèle : la compréhensibilité du modèle représente un enjeu majeur pour son adoption et son utilisation en conception. Nous avons donc préparé une seconde évaluation sous forme de questionnaire. Ce questionnaire a pour objectif d'évaluer la compréhensibilité du modèle. Nous présentons d'abord la première évaluation, une expérience de conception avec un « focus group », suivie de la seconde, le questionnaire sur la compréhensibilité du modèle.

3.2.1. Expérience de conception avec un groupe focalisé

Nous avons réalisé une expérience dans le cadre du projet ANR CARE (*Cultural experience: Augmented Reality and Emotion*[35], http://www.careproject.fr/). Cette expérience consistait à faire concevoir des techniques d'interaction avec un système de réalité mixte à un groupe d'acteurs de la conception (informaticiens spécialisés en Interaction Homme-Machine et ergonomes), afin d'observer les difficultés rencontrées dans l'utilisation du modèle. Les participants devaient concevoir l'interaction pour la découverte de la classification des espèces pour le musée d'histoire naturelle de Toulouse. Notre objectif pour cette expérience était d'étudier le rôle génératif du modèle au travers des productions émanant du groupe focalisé utilisant le modèle d'interaction mixte.

[35] Expérience culturelle : réalité augmentée et émotion

3.2.1.1. Protocole expérimental

Cinq personnes ont participé à cette expérience (2 femmes, 3 hommes) :
- Un médiateur (P1), seul spécialiste du modèle utilisé,
- un ergonome (P2),
- un informaticien (P3),
- un informaticien expert en IHM. (P4),
- un informaticien expert en interfaces mixtes (P5).

Deux observateurs ont également participé à l'expérience, pour assurer la prise de notes d'observation et la capture vidéo de la séance.

L'application retenue pour cette expérience était une installation muséale au musée d'histoire naturelle de Toulouse. La raison d'être du système est :
- d'inciter le visiteur à découvrir les différents caractères formant la classification des espèces,
- de lui faire comprendre qu'une espèce est un ensemble de caractères et que la complexité d'une espèce dépend du nombre de caractères évolués qui le composent,
- de l'aider à comprendre la structure de l'arbre de classification des espèces.

Pour simplifier notre cas d'étude pour l'expérience, nous avons choisi la classification des véhicules (motos, voitures, avions, etc.) à la place des espèces, trop nombreuses. L'arbre proposé pendant l'expérience est présenté à la Figure 139.

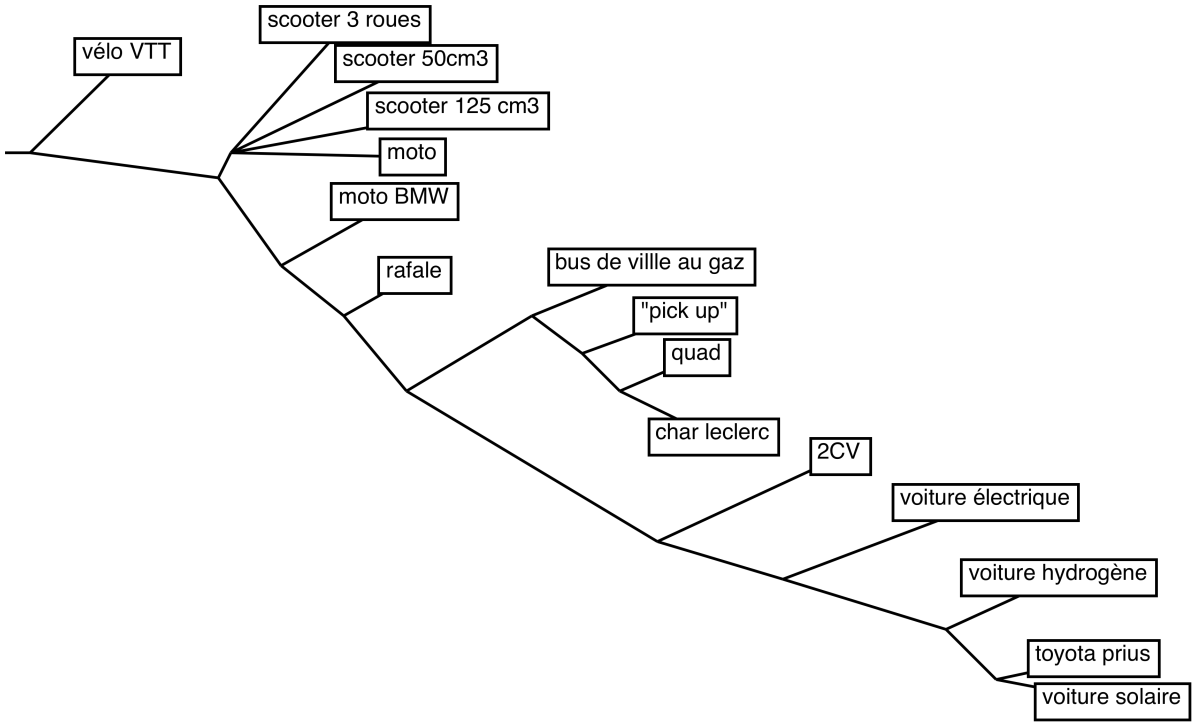


Figure 139 : Arbre de classification des véhicules, remplaçant les espèces pour l'expérience de conception en groupe focalisé.

La tâche retenue pour cette expérience était l'exploration de l'arbre de classification : Revenir au critère précèdent, aller au critère suivant, prendre connaissance des critères frères, etc. Un premier prototype de ce système existait déjà, et servait de point de départ à la séance de conception. Cette proposition initiale est présentée à la Figure 140. Ce prototype initial inclut deux points de vue de l'arbre :
- Une vue égocentrique présentée sur l'écran (Figure 140) : la présentation représente une pièce en 3D avec des portes qui représentent les critères suivants : les critères fils sont écrits sur les portes présentées en face.

- Une vue exocentrique sous forme papier (Figure 140) : la représentation papier de l'arbre correspond à une vue de dessus.

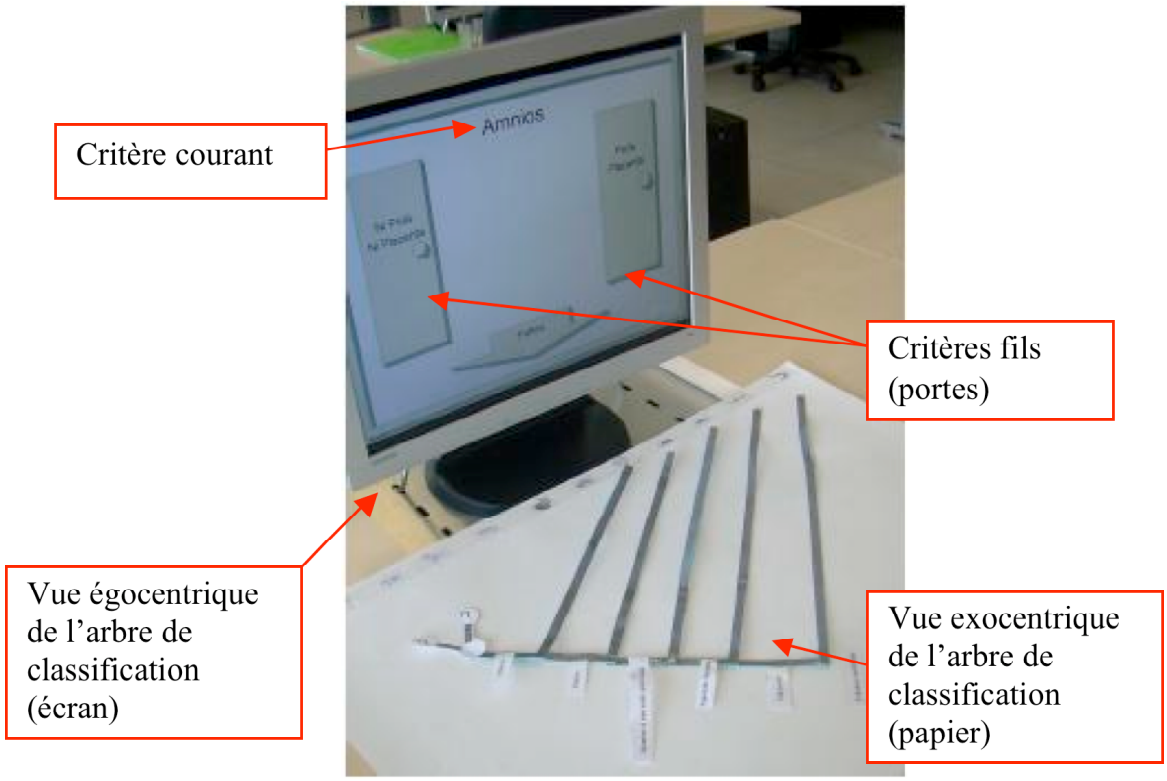


Figure 140 : La proposition initiale à partir de laquelle les participants au groupe focalisé ont conçu de nouvelles techniques d'interaction. Ce prototype propose deux points de vue de l'arbre : une vue égocentrique sur l'écran et une vue exocentrique sous forme papier.

L'expérience de conception a duré 2 heures. Le Tableau 22 présente le déroulement de l'expérience. Pendant une introduction de 30 minutes, le médiateur a présenté le contexte général de la séance (5 minutes), puis le modèle utilisé sur l'exemple d'ORBIS de la section 3.1.2.1 (15 minutes), et enfin le cas d'étude de la séance (10 minutes). La séance de génération de techniques d'interaction elle-même a duré une heure, la dernière demi-heure étant consacrée à une discussion visant à recueillir leurs commentaires sur le déroulement de l'expérience.

Introduction	Présentation générale du contexte de la séance	5 min
	Présentation du modèle utilisé	15 min
	Présentation du cas d'étude (principes et objectifs de la classification des espèces du vivant, description de la tâche, proposition initiale)	10 min
Génération de techniques d'interaction		60 min
Discussion		30 min

Tableau 22 : Déroulement de l'expérience avec un groupe focalisé sur la conception de techniques d'interaction.

3.2.1.2. Résultats

Le Tableau 23 présente les données relevées à partir des notes de l'observateur et de la vidéo. Nous avons relevé :

- Le temps avant l'utilisation du modèle par le médiateur (dès le début, puis à la 12ème minute la deuxième fois),

- Le temps avant utilisation du modèle par un autre membre que le médiateur (dès la deuxième minute, puis des éléments du modèle ont été explicitement évoqués à la troisième 3ème et à la 8ème minute).
- Le type de représentation des solutions générées (des modèles partiels, des dessins sketches de l'interface).

Les solutions conçues lors de cette expérience sont présentées en Annexe B. Outre ces données, les résultats de l'expérience au regard de nos objectifs résident dans les points de blocage rencontrés par les participants, et les améliorations qu'ils ont suggérées.

Temps avant l'utilisation du modèle par le médiateur	0 min	Évocation de l'objet de la tâche (Participant P1)
	~12 min	Reprise dans le modèle du design commencé (Participant P1)
Temps avant utilisation du modèle par un autre membre que le médiateur	~2 min	Participant P4 : Dessin de l'utilisateur
	~3 min	Participant P4 : « Donc là l'idée de penser la modalité d'interaction directement ? »
	~8 min	Participant P4 : « Est-ce ça fait partie de l'objet mixte, la visualisation de l'arbre ? »
Type de représentation des solutions générées	Modèles partiels, par l'expert du modèle (Participant P1) Sketchs de l'interface, dessinés par un participant principalement (Participant P4), parfois relayé par deux autres.	

Tableau 23: Données relevées à partir des notes de l'observateur et de la vidéo (P1 et P4 sont deux participants).

Nous avons observé un frein à l'utilisation du modèle : les participants non-experts du modèle y ont très peu fait référence. Pendant la discussion à la fin de l'expérience, les participants ont dit que le modèle avait trop de dimensions, qu'ils avaient eu des difficultés pour construire une solution à partir de la présentation du modèle. De plus, confirmant ces remarques, les participants ont suggéré les améliorations suivantes :

- Présenter le modèle plus simplement et intelligiblement pour des non-experts.
- Fournir un résumé simple du modèle, sur papier, comme outil de travail à utiliser pendant la séance.
- Animer la séance en posant des questions aux participants pour les forcer à explorer le modèle, mais ne pas leur demander de le manipuler.
- Ne pas présenter tout le modèle d'un coup au début suivi d'un brainstorming trop libre, mais structurer la séance en étapes.
- Ne pas utiliser de termes du domaine pour les non-experts (comme « objet de la tâche »).
- Choisir entre le rôle de médiateur et de participant pour l'expert du modèle, mais pas les deux.
- Présenter très rapidement les bases du modèle, puis au cours de la conception, faire des pauses pour expliciter la solution avec le modèle et reprendre la conception en ayant proposé des possibilités grâce au modèle (suggéré après l'expérimentation par quelqu'un n'ayant pas participé).

De ces remarques, nous retenons l'importance de présenter le modèle de façon adéquate aux participants. Le modèle était clairement pas assez compris pour être utilisé. De plus, les résultats de l'expérience étaient fortement liés à l'application choisie pour la conception. Celle-ci s'inscrivait parfaitement dans le cadre du projet ANR CARE (http://www.careproject.fr/), mais elle a paru trop compliquée : les participants nous ont suggéré de choisir une application plus simple. Pour les prochaines expériences de conception en groupe focalisé dans le cadre du projet CARE, il convient de mettre au point une présentation du modèle sans doute introduite progressivement lors des expériences.

En conclusion, lors de cette expérience, nous n'avons que partiellement évalué le pouvoir génératif du modèle d'interaction mixte. Comme dans l'expérience avec Yubi, il nous a semblé que l'enjeu principal est la compréhensibilité du modèle. Nous avons donc étudié, à la suite de ces expériences, la compréhensibilité du modèle auprès de concepteurs.

3.2.2. Évaluation exploratoire de la compréhensibilité du modèle

La compréhensibilité du modèle représente un enjeu majeur pour son adoption et son utilisation en conception. Nous avons donc évalué la compréhensibilité du modèle d'interaction mixte. Afin de mener cette évaluation, nous nous sommes inspirés d'une méthode d'évaluation des modèles logiciels [Anranda et al, 2007]. En effet, les modèles logiciels (par exemple UML) comme les modèles d'interaction servent à définir une représentation d'un artefact qui doit être partagée et qui doit permettre la communication entre les acteurs. La compréhensibilité de ces modèles est donc essentielle à son adoption.
Nous présentons d'abord le protocole expérimental de cette évaluation de la compréhensibilité du modèle, puis les résultats issus de cette évaluation.

3.2.2.1. Protocole expérimental

Pour évaluer la compréhensibilité du modèle d'interaction mixte :

1. Nous avons fixé la **notation utilisée** lors de l'expérience : les modèles utilisés dans le questionnaire ont été produits par un expert. De plus, quelques ajustements sur la notation ont été autorisés, comme cela se produit inévitablement lors d'une utilisation réelle en conception (annotations en langage naturel à côté des schémas, dessins, etc.). Pourtant, les schémas produits n'étaient pas dessinés à main levée, afin de ne pas évaluer l'écriture de l'expert, mais bien la notation. Les questionnaires contenant ces schémas sont présentés en Annexe C.
2. Nous avons fixé une **situation pertinente** pour l'expérience : les utilisateurs cibles de la notation sont tous ceux qui peuvent participer à la conception d'une interface mixte. Nous avons donc choisi 18 sujets, ayant une expertise allant de l'informatique et l'Interaction Homme-Machine (6 sujets), jusqu'au design (arts appliqués) (6 sujets), en passant par des ergonomes (6 sujets). Nous avons choisi pour l'évaluation de proposer des modèles de l'interaction avec des applications très grand public (une application de bureau et de photos), afin de ne pas introduire la variable « expertise du domaine d'application » dans l'évaluation. Pour aller plus avant dans cette évaluation de la compréhensibilité, il conviendra à l'avenir d'évaluer la compréhensibilité en fonction de cette variable. De même, nous avons fixé les autres variables qui influent sur la compréhensibilité :
    a. le type de tâche (compréhension d'un modèle complet produit en conception),
    b. l'expertise du modèle qu'ont les utilisateurs (nulle),
    c. la taille du problème (les applications choisies).
3. Nous avons fixé les **variables mesurées** : nous avons mesuré les variables sur lesquelles la compréhensibilité influe :
    1. l'exactitude/justesse de ce qui a été compris,
    2. le temps requis perçu (subjectif) pour comprendre la représentation,
    3. la confiance (subjective) que les utilisateurs ont dans leur propre compréhension du modèle,
    4. la difficulté perçue (subjective) que les utilisateurs ont eue pour accéder à l'information contenue dans les schémas.
4. Nous avons fixé une **référence** pour comparaison (pour que les résultats soient indépendants de la complexité du domaine utilisé lors de l'expérience) : Nous avons choisi ASUR [Dubois, Gray, 2007] comme notation de référence. ASUR est différent du modèle évalué, mais est capable de couvrir des besoins similaires pendant la conception.

Nous avons conçu deux questionnaires, présentés en Annexe C, afin de contrebalancer les différentes variables (Tableau 24) : expertises des utilisateurs, applications exemples, et modèles utilisés. Les sujets ont été divisés en deux groupes A et B. Chaque groupe contenait un nombre égal d'informaticiens, de designers et d'ergonomes.

	Premier exemple de système : le bureau	**Deuxième exemple de système : les photos**
Groupe A : 3 informaticiens 3 designers 3 ergonomes	Modèle d'Interaction Mixte	ASUR
Groupe B : 3 informaticiens 3 designers 3 ergonomes	ASUR	Modèle d'Interaction Mixte

Tableau 24 : Contrebalancer les variables dans l'évaluation de la compréhensibilité du modèle d'interaction mixte.

De plus, nous avons ajouté des questions d'ordre qualitatif, relevant de la compréhension du modèle d'interaction mixte. Ces questions sont présentées de façon identique à la fin des deux questionnaires. Ces questions nous permettent d'évaluer plus précisément la compréhension du modèle et d'obtenir des informations essentielles pour améliorer la compréhensibilité du modèle. Les questionnaires ont été donnés aux sujets en main propre, ou envoyés par courrier électronique ou postal.
Nous sommes conscients du biais introduit par le fait que le questionnaire n'a pas été rempli par les sujets informaticiens/designers/ergonomes en présence d'un expérimentateur. De plus l'échantillon choisi (18 personnes, 6 de chaque profession) est limité, et ne nous permettra pas de *prouver* la compréhensibilité du modèle. Néanmoins, les résultats obtenus grâce à cette évaluation exploratoire permettent d'améliorer le modèle d'interaction mixte, mais aussi de concevoir une nouvelle enquête.

3.2.3. Résultats

Sept femmes et onze hommes ayant entre 23 et 42 ans ont participé à l'enquête. Parmi eux, 6 informaticiens, 6 designers et 6 ergonomes. Ils avaient entre 0 (étudiant) et 14 ans d'expérience dans leur métier et ont mis en moyenne une heure pour répondre au questionnaire.

Nous présentons tout d'abord les résultats sur les différentes dimensions de la compréhensibilité : exactitude de la compréhension, facilité, rapidité, confiance et note de compréhensibilité.

Pour évaluer l'exactitude de la compréhension des modélisations, nous avons notés les concepts attendus dans les explications par les sujets des modélisations. Nous avons noté les concepts bien identifiés, les concepts non précis (c'est-à-dire bien identifiés mais manquant de précision), les concepts mal identifiés (c'est-à-dire identifiés mais de façon erronée), ainsi que les concepts non identifiés (absents de la réponse). La Figure 141 montre pour le modèle d'interaction mixte la proportion des concepts attendus selon ces quatre classes. Nous soulignons ici que peu de sujets ont fait de fautes, mais qu'une majorité des concepts véhiculés par le modèle ne sont pas identifiés.

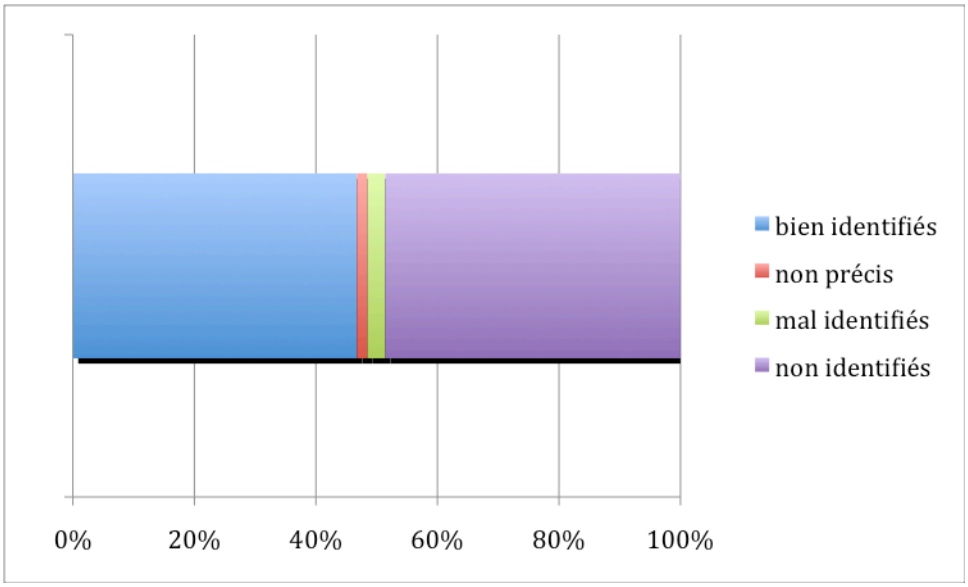


Figure 141 : Exactitude de la réponse pour le modèle d'interaction mixte : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés.

La Figure 142 affine la précédente selon le profil des sujets : Comme pour ASUR (Figure 143), les ergonomes et les designers ont plus difficilement compris les modélisations proposées. Nous notons néanmoins que les sujets ont manquer davantage de précision avec ASUR (Figure 143 en rouge) qu'avec le modèle d'interaction mixte (Figure 142 en rouge).

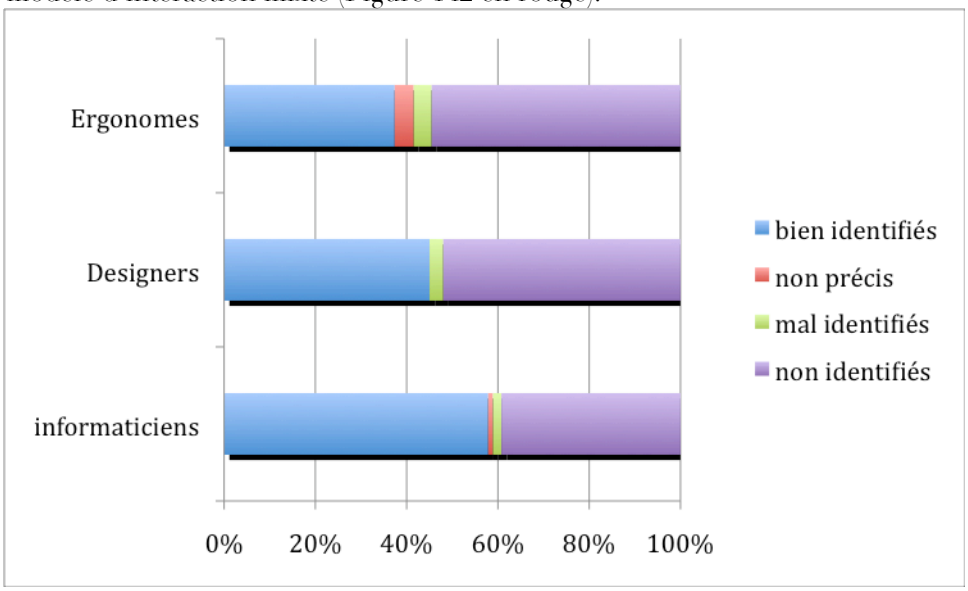


Figure 142 : Exactitude de la réponse pour le modèle d'interaction mixte selon le profil des sujets : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés.

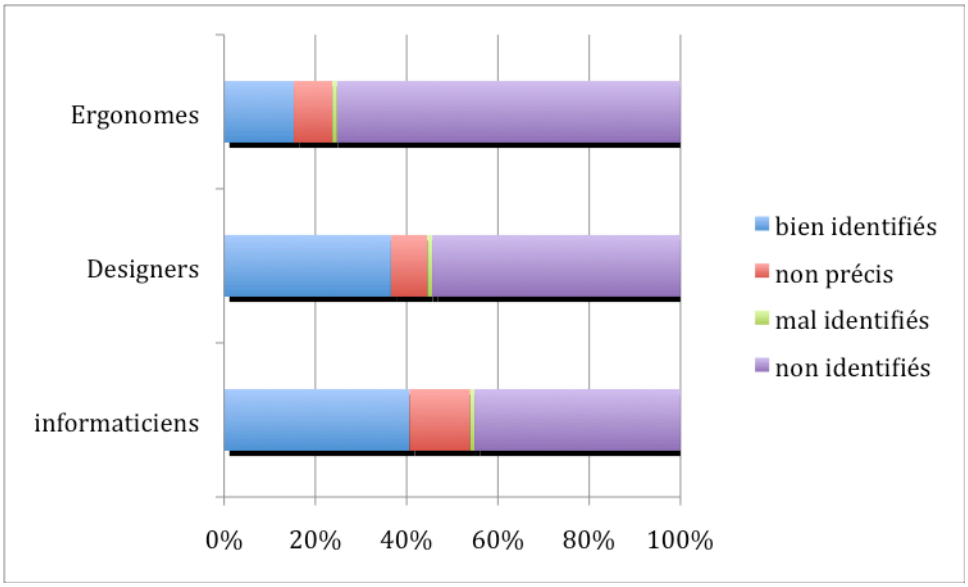


Figure 143 : Exactitude de la réponse pour ASUR selon le profil des sujets : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés.

Nous présentons maintenant les résultats pour les variables subjectives mesurées. Tout d'abord, en moyenne, les participants ont affirmé avoir expliqué facilement les modèles d'interaction mixte (contre difficilement en moyenne pour les modèles ASUR). La Figure 144 montre la répartition de la facilité moyenne pour chaque modèle selon les profils de sujets.

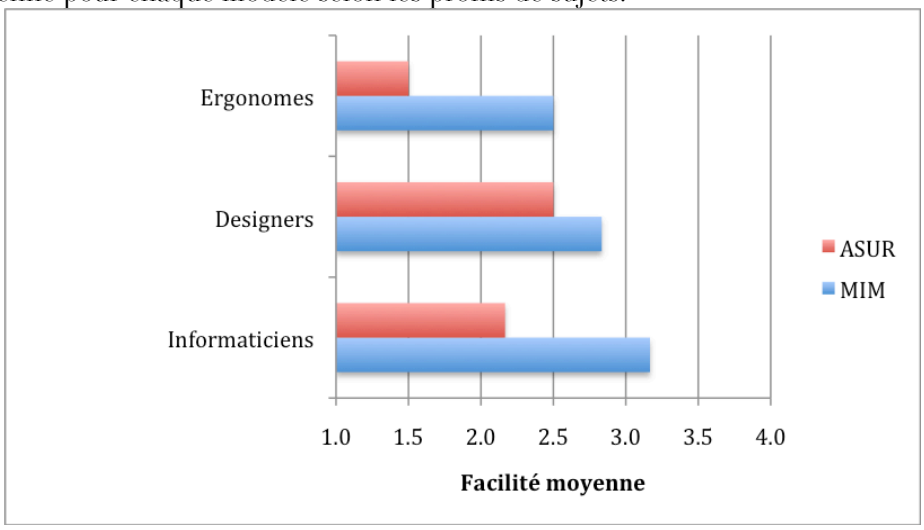


Figure 144 : Facilité moyenne (subjective) selon les profils (1 signifie très difficile, 4 signifie très facilement).

Ils ont aussi affirmé, en moyenne, avoir expliqué rapidement les modèles d'interaction mixte (contre lentement en moyenne pour les modèles ASUR). La Figure 145 montre la répartition du temps moyen pour chaque modèle selon les profils de sujets.

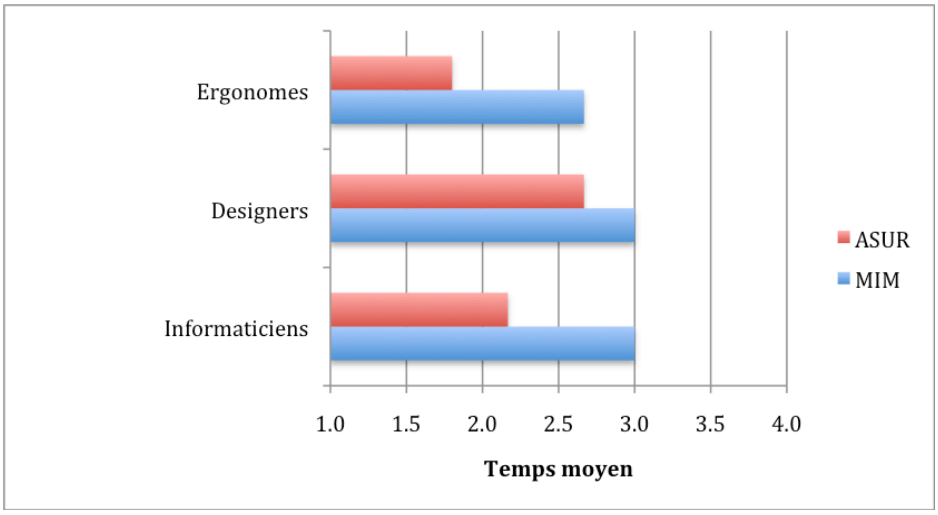


Figure 145 : Temps moyen (subjectif) selon le profil et le modèle (1 signifie très lentement, 4 signifie très rapidement).

La moyenne des notes de confiance accordée à leur réponse est de 6,3/10 pour le modèle d'interaction mixte, contre 4,7/10 pour ASUR. La Figure 146 montre la répartition des notes de confiance pour ASUR et le modèle d'interaction mixte selon les profils. Nous soulignons ici que même si les moyennes pour le modèle d'interaction mixte est meilleure que pour ASUR pour cette expérience, la répartition (Figure 146) souligne néanmoins que la confiance des participants dans ce qu'ils ont compris est globalement moyenne.

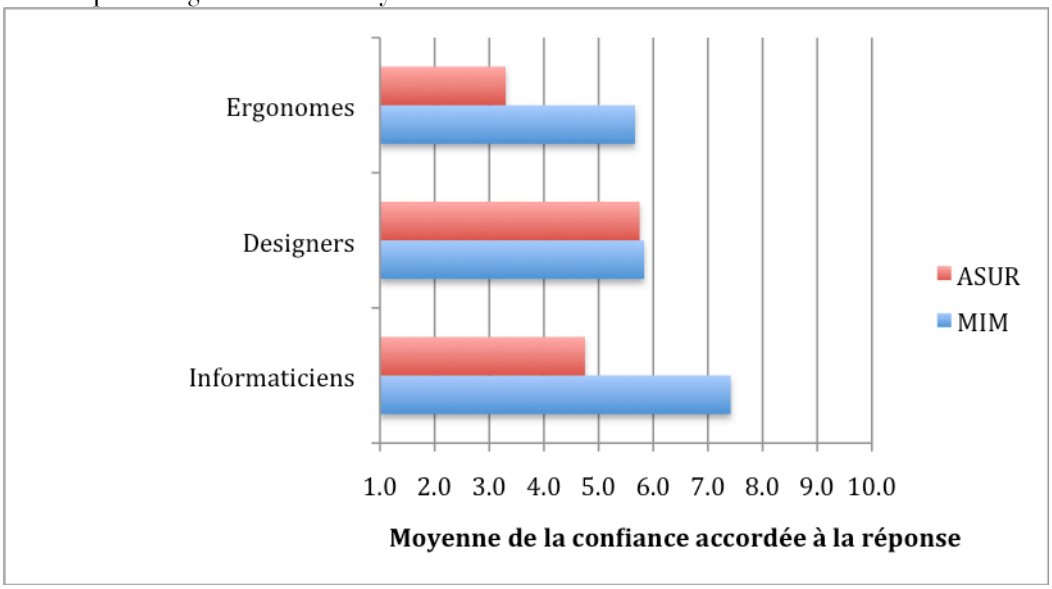


Figure 146 : Confiance moyenne accordée à la réponse (1 signifie pas confiant du tout, 10 signifie tout à fait confiant).

La moyenne des notes de compréhensibilité attribuée par les participants aux schémas est de 5,4/10 pour le modèle d'interaction mixte, contre 3,3/10 pour ASUR. La Figure 147 montre la répartition des notes de compréhensibilité attribuée aux modèles par les sujets selon leur profil. Pour cette note, comme pour la facilité (Figure 144), le temps (Figure 145) et la confiance (Figure 146) : l'écart entre le modèle d'interaction mixte et ASUR plus grand pour les informaticiens et les ergonomes que pour les designers.

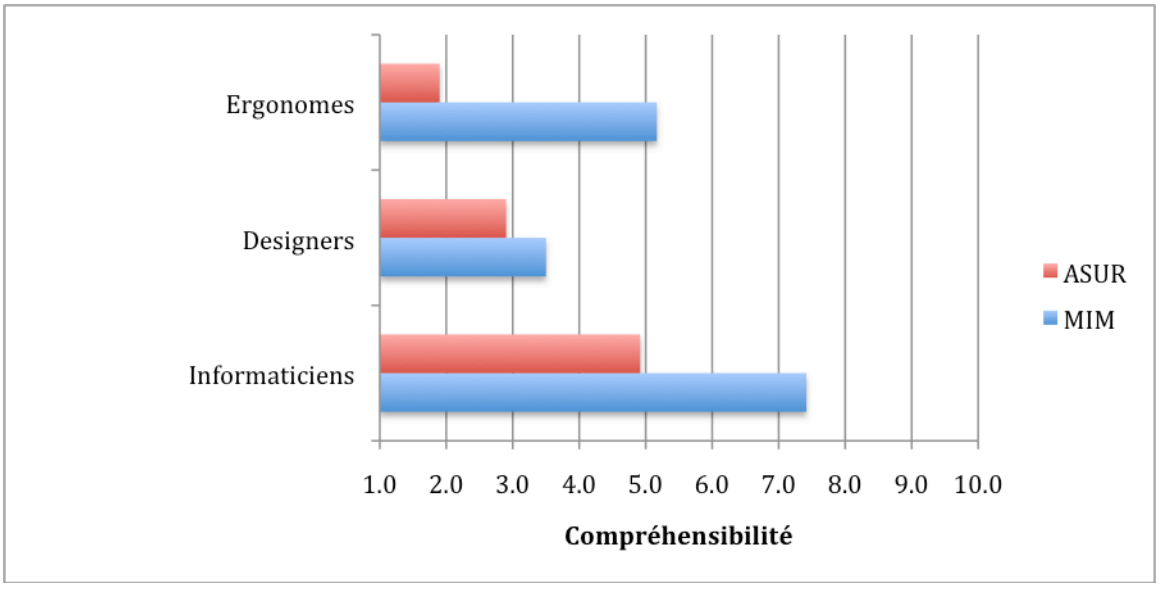


Figure 147 : Moyenne des notes de compréhensibilité des schémas par tous les sujets pour le modèle d'interaction mixte et pour ASUR.

La Figure 149 montre la répartition des notes de compréhensibilité pour ASUR et le modèle d'interaction mixte. Là encore, nous soulignons que même si la moyenne pour le modèle d'interaction mixte est meilleure que pour ASUR pour cette expérience, la répartition (Figure 149) montre que les participants trouvent le modèle d'interaction mixte peu compréhensible.

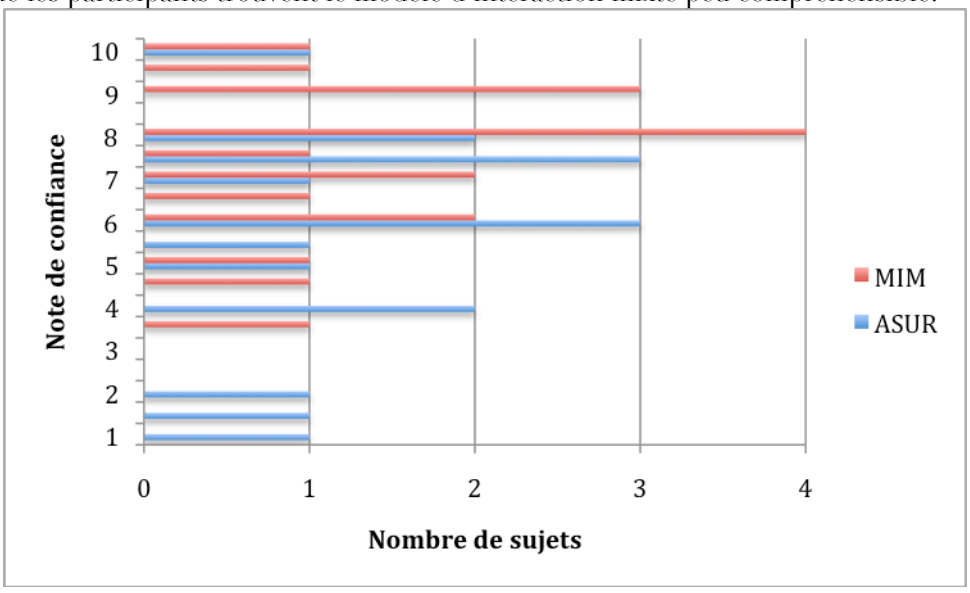


Figure 148 : Effectif des participants en fonction des notes de confiance attribuées (10 signifie tout à fait confiant et 1 pas du tout confiant).

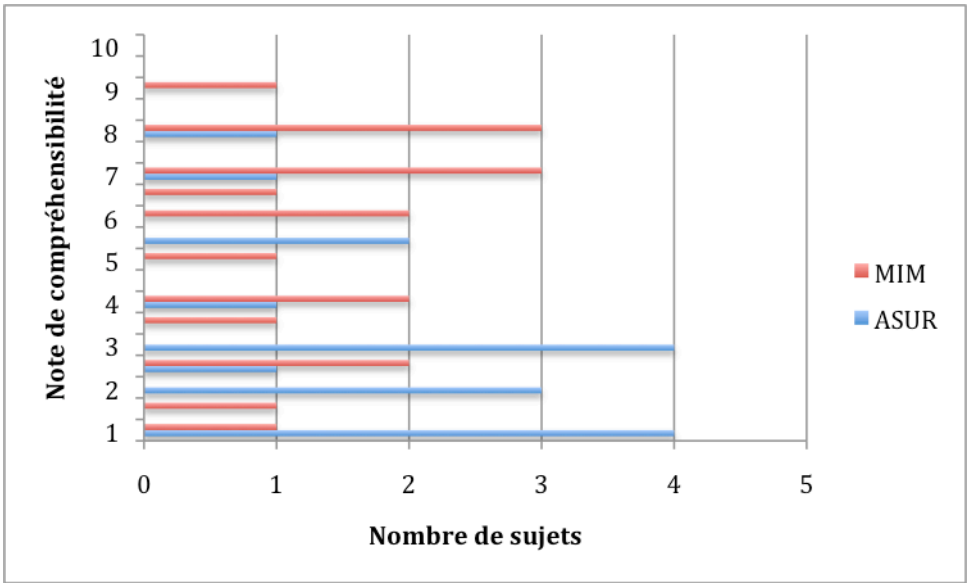


Figure 149 : Effectif des participants en fonction des notes de compréhensibilité attribuées (10 signifie tout à fait compréhensible et 1 pas du tout compréhensible).

Pourtant, venant confirmer les résultats encourageants du modèle d'interaction mixte, 11 personnes sur 18 (61% des participants) ont déclaré qu'ils auraient choisi le modèle d'interaction mixte s'ils devaient décrire un système interactif. Parmi les autres participants, 6 ont déclaré qu'ils n'auraient choisi aucun des deux schémas pour décrire un système interactif. Ces résultats suggèrent que la majorité des participants ont été convaincus par le modèle d'interaction mixte, même si une proportion non négligeable (1/3) n'ont pas été convaincue et nous incite à être prudents.

Nous présentons maintenant les résultats sur la compréhension des éléments constitutifs du modèle d'interaction mixte. Nous remarquons tout d'abord que la représentation graphique des propriétés physiques a été globalement une aide pour la compréhension. Nous soulignons aussi grâce à la Figure 150 que l'écart d'appréciation entre icônes symboliques (ASUR) et représentations figuratives (MIM) se creuse des informaticiens aux ergonomes en passant par les designers. Onze participants déclarent qu'ils préfèreraient avoir les propriétés physiques sous forme textuelle et graphique (les deux représentations). Les représentations multiples sont donc à considérer dans le modèle.

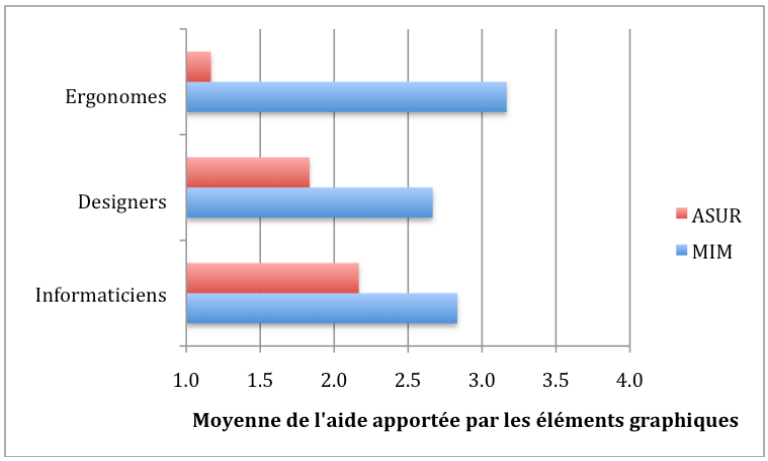


Figure 150 : Moyenne par profil de l'aide apportée par les choix graphiques des deux notations : 1 signifie « n'aide pas du tout », 4 signifie « aide tout à fait ».

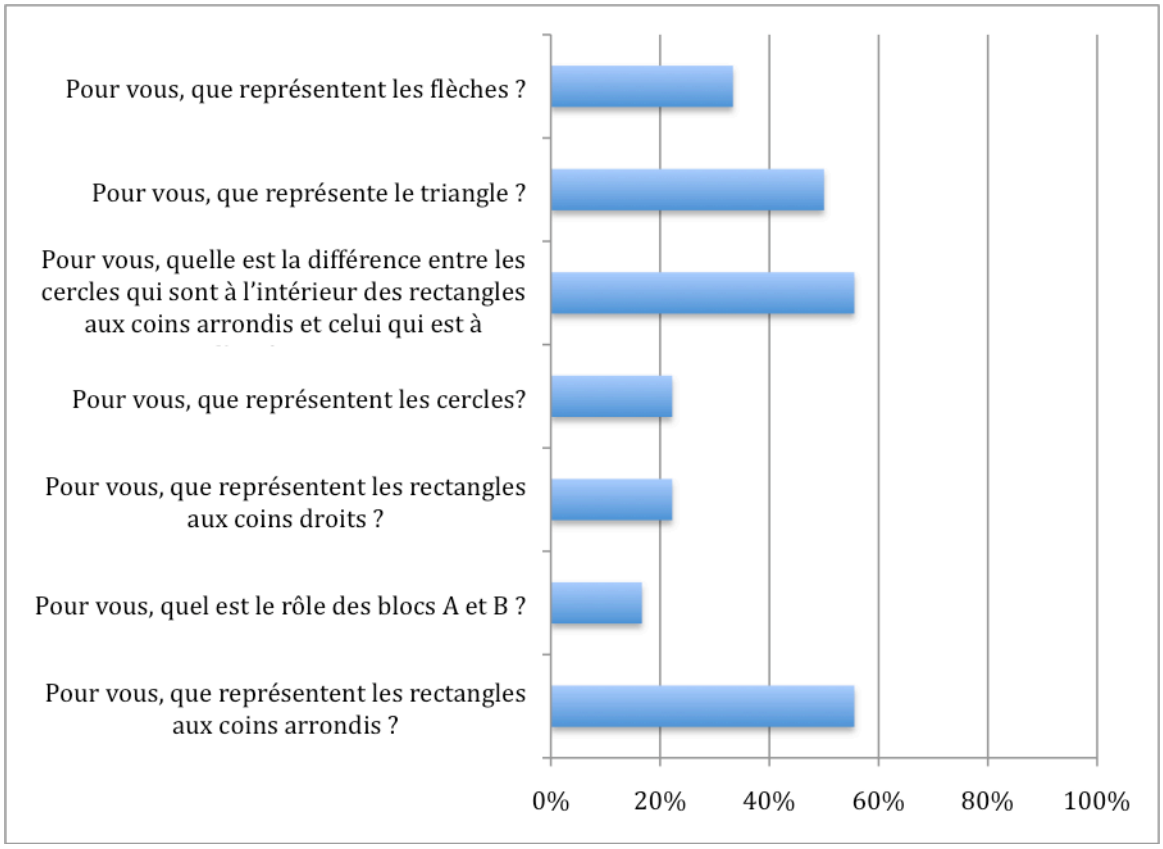


Figure 151 : Proportion de bonnes réponses aux sept questions de compréhension des éléments du modèle d'interaction mixte.

La Figure 151 présente la proportion de bonnes réponses aux sept questions de compréhension des éléments du modèle d'interaction mixte. La première, cinquième et sixième questions (en partant du bas de la Figure 151) concernent respectivement les objets mixtes, les langages d'interaction et la composition des modalités de liaison :

- « Pour vous, que représentent les rectangles aux coins arrondis ? » (des objets à moitié physique et à moitié numérique).
- « Pour vous, quelle est la différence entre les cercles qui sont à l'intérieur des rectangles aux coins arrondis et celui qui est à l'extérieur ? » (ceux qui sont à l'intérieur font le lien entre les parties physiques et numériques de l'objet, ceux qui sont à l'extérieur représentent l'interaction entre les objets).
- « Pour vous, que représente le triangle ? » (La décomposition/séparation/fission de l'information).

À ces trois questions, une majorité de participants a bien répondu (50 à 56%), mais nous constatons que c'est une faible majorité.

À la question « Pour vous, que représentent les flèches ? » (des flux d'informations), seuls 33% des participants ont bien répondu. Aux questions suivantes, seuls 22% des participants ont bien répondu :

- « Pour vous, que représentent les rectangles aux coins droits ? » (Des parties physiques ou numériques),
- « Pour vous, que représentent les cercles ? » (une transformation).

Enfin, seuls 17% des participants ont bien répondu à la question « Pour vous, quel est le rôle des blocs A et B ? » (A est un outil et B est l'objet cible de l'utilisateur). Ceci montre qu'il est nécessaire de travailler la représentation du modèle d'interaction mixte, puisque l'une de ses différences avec ASUR (l'encapsulation des objets d'interaction) n'est pas comprise.

Grâce à ces résultats, nous pouvons maintenant (1) travailler à une meilleure présentation des concepts mal compris et (2) concevoir une nouvelle évaluation pour mieux comprendre ce qui induit en erreur les participants pour chaque notion mal comprise.
Tout d'abord, cette évaluation met en exergue un besoin de travailler sur la représentation, non seulement pour sa compréhensibilité, mais aussi à l'avenir pour son pouvoir d'expression. En effet, comment exprimer directement dans un schéma clair et compréhensible la vingtaine de caractéristiques du modèle d'interaction mixte ? Plusieurs pistes pour améliorer la notation sont à considérer, dont la présentation du modèle selon un schéma non académique, comme les méthodes de conception basées sur le jeu présentées dans [Kultima, 2008].
Ensuite, il sera possible de remédier aux faiblesses du protocole de cette première expérience. Tout d'abord, nous proposerons à plus de personnes de participer pour pouvoir confirmer les tendances identifiées dans cette étude. Ensuite, nous proposerons d'effectuer cette expérience en présence d'un expérimentateur. Ceci nous permettra de lier plus facilement les résultats de l'expérience avec les différences entre les modèles grâce aux commentaires (qualitatifs) des participants. De plus, nous pourrons mesurer d'autres dimensions que la compréhensibilité, qui n'est qu'une facette de la facilité d'utilisation. Enfin, les prochaines expériences pourront cibler des aspects particuliers du modèle, en comparant par exemple les résultats de conception de deux groupes d'utilisateur (avec ou sans modèle) et en les questionnant sur les alternatives qu'ils ont envisagées selon une dimension particulière.

4. Synthèse

Nous avons réalisé des évaluations conceptuelles et expérimentales, avec protocole ou non, pour évaluer le pouvoir génératif du modèle d'interaction mixte. Les évaluations conceptuelles nous ont permis d'estimer ses qualités et défauts au plus tôt. Les premières études expérimentales, réelles et sans protocole, constituent le carnet de route de l'utilisation du modèle d'interaction mixte. Elles nous ont permis d'améliorer son pouvoir génératif, en faisant évoluer le modèle par l'ajout de caractéristiques intrinsèques et extrinsèques à un objet mixte. Au fil de ces expériences, tandis que le modèle s'est enrichi de caractéristiques pour obtenir une version stable, nous avons aussi fait évoluer l'expertise des participants aux expériences : d'informaticiens experts du modèle et en Interaction Homme-Machine, nous avons ensuite considéré l'utilisation du modèle par des non experts comme un designer (arts appliqués). Ce passage progressif nous a permis d'identifier les difficultés rencontrées par les utilisateurs non experts. Afin de mieux cerner ces difficultés, nous avons mené des évaluations avec protocole.
De toutes ces expériences sont ressortis les avantages suivants : le modèle d'interaction mixte nous a permis lors des expériences d'explorer l'espace de conception en nous reposant sur les caractéristiques du modèle (section 2 du Chapitre 4) dont certaines sont issues de travaux existants. Il nous a permis d'amorcer la conception par une approche systématique. Le modèle a été un support à la conception tout en permettant d'intégrer des solutions partielles qui n'ont pas nécessairement été conçues grâce au modèle : une fois décrites avec le modèle, nous avons pu continuer l'exploration systématique à partir de ces solutions partielles.
Néanmoins ces expériences nous ont aussi permis de toucher du doigt un enjeu important d'un modèle qu'est sa compréhensibilité par des non experts. Les perspectives à court terme pour ce travail d'évaluation du pouvoir génératif du modèle d'interaction mixte incluent une meilleure présentation du modèle à des non experts et la conduite d'expériences en groupe focalisé (dans le cadre du projet CARE). De plus la compréhensibilité du modèle n'est qu'une facette de son pouvoir génératif. Un modèle d'interaction peut être tout à fait compréhensible, mais difficile à utiliser pour explorer l'espace de conception. Aussi une version étendue du questionnaire pour considérer la production de modèles a été proposée à des étudiants de Master Informatique spécialité IHM.
Tout au long de ces expériences sur des cas de conception de systèmes de réalité mixte, nous avons focalisé sur l'utilisation du modèle et des modélisations résultantes en ne décrivant pas sciemment les prototypes développés au cours de ces expériences. Or, le prototypage ne peut être dissocié de la modélisation en phase de conception. Aussi tandis que les Chapitres 3, 4 et 5 ont focalisé sur la modélisation, les trois chapitres suivants, selon une démarche similaire, sont dédiés au prototypage.

De la modélisation conceptuelle au prototypage

Dans la Partie I (Chapitres 3, 4 et 5), nous avons focalisé sur l'activité conceptuelle de conception d'interfaces mixtes. Nous avons tout d'abord présenté un état de l'art des outils conceptuels pouvant aider la conception et nous avons souligné leurs apports et limites (Chapitre 3). Nous avons ensuite présenté un modèle de l'interaction avec les systèmes de réalité mixte (Chapitre 4). Nous avons finalement présenté le carnet de route de l'élaboration itérative du modèle et les différentes formes d'évaluation conduites de notre modèle d'interaction mixte (Chapitre 5).

Lors de la présentation d'expériences de conception réelles, concrètes et sans protocole dans le cadre de l'étude du pouvoir génératif du modèle d'interaction mixte, nous avons volontairement focalisé sur la modélisation conceptuelle et omis les prototypes réalisés pendant ces séances. Or les prototypes réalisés par une équipe de concepteurs sont des outils essentiels pour la communication, la comparaison et l'exploration de l'espace de conception [Lim et al., 2008]. L'exploration rationnelle et systématique de l'espace de conception ne peut pas être effectuée de façon réaliste par la seule activité de modélisation conceptuelle, comme le montre le cycle de la Figure 152. Outre leur rôle dans l'identification précoce des problèmes et erreurs potentiels, les prototypes servent aussi de supports à la comparaison et à la génération des alternatives. Les activités pratiques (assistées par le prototypage) et conceptuelles (assistées par le modèle de l'interaction mixte) sont inextricables : « […] *design is a continuous coupling of internal mental activities and external realization activities* » [Lim et al., 2008]. De nombreuses études confirment ces pratiques [Avrahami, Hudson, 2002][Buxton, 2007][Hartmann et al., 2008][Gaver, Martin, 2000]. Puisque la conception est une activité qui est réalisée en pratique en prototypant, il est donc essentiel d'élargir l'aide à la conception fournie par notre modèle au champ du prototypage. L'objectif est d'opérationnaliser le modèle d'interaction mixte présenté au Chapitre 4 afin de mieux articuler exploration conceptuelle et pratique.

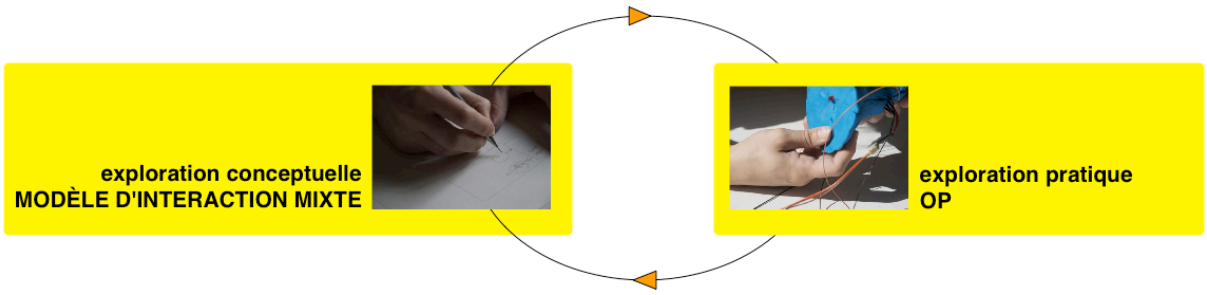


Figure 152 : La Partie I du mémoire présente notre contribution concernant l'exploration conceptuelle de l'espace des possibilités (à gauche, le modèle d'interaction mixte) et la Partie II présente notre contribution concernant l'exploration pratique de l'espace des possibilités (à droite, l'outil logiciel OP).

Tandis que la Partie I (Chapitres 3, 4 et 5) a focalisé sur les aspects de modélisation en présentant un état de l'art, notre contribution puis son élaboration et évaluation, nous structurons la Partie II (Chapitres 6, 7 et 8) dédiée au prototypage de façon similaire. Le Chapitre 6 présente un état de l'art des outils de prototypage, le Chapitre 7 présente notre contribution, puis le Chapitre 8 présente son évaluation.

Partie II : Prototypage

Chapitre 6 : État de l’art des outils de prototypage

Ce chapitre présente un état de l'art des outils de prototypage pour les interfaces mixtes. Avant de dresser cet état de l'art, il convient de mieux définir cette activité. Selon les auteurs de [Lim et al., 2008], les prototypes permettent d'explorer les possibilités de conception concernant différentes **dimensions**, comme par exemple l'apparence (les propriétés physiques), l'interactivité, ou l'arrangement spatial. Les prototypes sont réalisés par les concepteurs afin d'explorer certaines de ces dimensions.
Les concepteurs réalisent alors des prototypes en fonction de leurs besoins pour explorer ces dimensions, avec des **matériaux**, **envergure**, et **résolution** particuliers :

- Le **matériau** utilisé peut être du papier, du bois, du plastique, des objets déjà manufacturés, mais aussi produit par un programme Macromedia® Flash® et/ou fait avec des capteurs et des effecteurs comme les Phidgets de [Greenberg, Fitchett, 2001].
- L'**envergure** d'un prototype est l'ensemble des dimensions explorées grâce à ce prototype. Par exemple, un prototype peut matérialiser des idées concernant l'interaction, mais pas l'apparence, ou les deux (il a dans ce cas une plus grande envergure).
- Un prototype peut être à très basse définition/**résolution**, avec très peu de détails et très peu fidèle au design final. Au contraire, le prototype peut être très proche du design final. Les concepteurs peuvent par exemple réaliser un prototype dont la résolution de l'interaction est très basse alors que la résolution de l'apparence est très haute. L'intérêt des prototypes à basse résolution est multiple, de la communication entre les concepteurs pendant leur travail à la richesse des suggestions faites lors de tests avec des utilisateurs [Landay, 1996].

Les choix des matériaux, de la résolution et de l'envergure sont faits en fonction du coût (temps, matériel, expertise). Un prototype doit permettre de matérialiser et d'explorer les dimensions voulues, avec un minimum d'effort de réalisation. La résolution et l'envergure des prototypes réalisés augmentent en général au fur et à mesure de la conception jusqu'au design final [Avrahami, Hudson, 2002].

Le prototypage est pratiqué de cette façon, par des professionnels, de l'informatique au design, en passant par d'autres professions comme les ergonomes, pendant les séances de conception. Lors de séances de conception participative [Mackay, 1996], les utilisateurs finaux (amateurs) participent aussi à la conception. Cette tendance s'inscrit dans une pratique plus large qu'on pourrait appeler le « DIY » ou « do-it-yourself », où des concepteurs d'interfaces ont un statut allant d'amateur à professionnel en passant par amateur éclairé, avec un éventail de compétences très diverses. Il s'agit d'une pratique populaire parallèle à la production professionnelle. De l'édition généraliste comme le magazine MAKE (http://www.makezine.com), cette pratique a des échos maintenant en recherche [Hartmann et al., 2008][Lee, 2008]. Ces pratiquants de la conception d'interfaces aspirent à savoir réaliser leurs propres solutions conçues. Des outils pour le prototypage comme Arduino (http://www.arduino.cc/) sont utilisés par ces concepteurs. Cette tendance croise par certains aspects un axe de recherche en Interaction Homme-Machine appelé « programmation par l'utilisateur final[36] ».
Outre les liens de certains outils de prototypage que nous présentons avec les approches de programmation par l'utilisateur final, les outils présentés concernent aussi d'autres axes de recherche, comme les interfaces mobiles ou les interfaces multimodales. Dans ce chapitre, nous analysons ces outils, au regard de nos propres objectifs. Nos objectifs, que nous présentons ci-dessous, concernent le prototypage des interfaces mélangeant éléments physiques et numériques.

1. Objectifs

L'objectif de l'outil visé se situe à deux niveaux :

1. En considérant le prototypage de façon isolée, notre objectif est de permettre le prototypage rapide de l'interaction, avec un maximum de résolution et d'envergure.

[36] end-user programming

2. En considérant le prototypage comme l'aspect matérialisation de la conception, notre objectif est d'opérationnaliser le modèle d'interaction mixte présenté au Chapitre 4 afin de permettre une articulation efficace et fluide entre exploration conceptuelle et pratique.

1.1. Pour le prototypage isolé

Nous détaillons ce premier objectif en quatre sous-objectifs :

- Être capable de prototyper très tôt à la fois l'interaction accompagnée de l'apparence, pour augmenter l'envergure des prototypes. En effet, comme souligné dans [Hudson, Mankoff, 2006], jusqu'ici, les concepteurs pouvaient réaliser des prototypes papier d'interface graphiques pour prototyper à la fois l'apparence et l'interaction. Pour les interfaces mixtes, qui prennent d'autres formes que des fenêtres sur un écran, les prototypes interactifs se limitent aux dispositifs aux formes fixes tels que les téléphones ou les PDAs. Les concepteurs se retrouvent donc à réaliser des prototypes de l'apparence d'un côté (par exemple en pâte à modeler) et des prototypes de l'interaction de l'autre côté (avec une simulation sur l'écran par exemple). Le processus de développement demande alors de faire des allers-retours entre forme et interaction pour arriver au système final (Figure 153), ce qui demande du temps et des efforts aux concepteurs. Nous voudrions réunir les premiers prototypes afin que l'apparence et l'interaction, qui s'inter-influencent, puissent être prototypées en même temps.

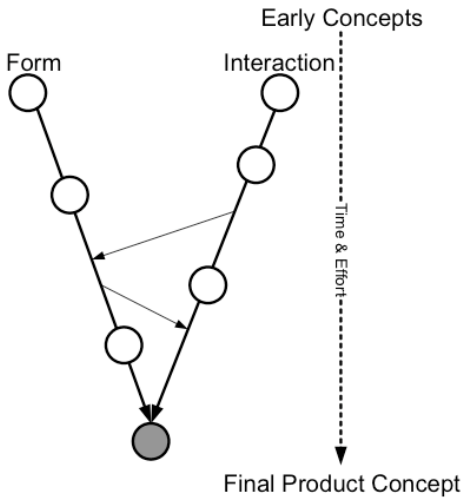


Figure 153 : Prototypage simultané mais séparé de l'apparence et de l'interaction. Illustration extraite de [Hudson, Mankoff, 2006].

- Baisser le coût du prototypage de l'interaction, pour augmenter la résolution des premiers prototypes.
- Permettre une évolution rapide pour faciliter les modifications dans le cadre de micro-itérations.
- S'adapter aux innovations technologiques existantes et être capable de s'adapter aux innovations à venir.

D'un point de vue logiciel, ces objectifs pour notre outil peuvent être mis en correspondance avec les critères suivants de qualité logicielle (critères de MacCall et norme ISO 9126), appliqués à un outil de prototypage :

- La maintenabilité : Il s'agit de l'effort pour effectuer des modifications dans le prototype : corriger les défauts, atteindre de nouveaux objectifs, faire face à un changement d'environnement.
- La compatibilité : Il s'agit de la facilité avec laquelle un logiciel peut être combiné avec d'autres logiciels.

- La portabilité : Il s'agit de la facilité avec laquelle un logiciel peut être transféré sous différents environnements matériels et logiciels.
- La réutilisabilité : Il s'agit de l'aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.
- L'extensibilité : Il s'agit de la facilité avec laquelle un logiciel se prête à une modification ou à une extension des fonctions qui lui sont demandées.

Outre ces objectifs intrinsèques à notre outil logiciel, nous définissons d'autres objectifs en considérant le contexte de conception avec le modèle d'interaction mixte.

1.2. Pour l'aspect prototypage de la conception

En repositionnant le prototypage dans la conception comme nous l'avons fait dès l'introduction (Chapitre 1), le prototypage n'est plus une activité isolée. En effet, il permet la matérialisation des solutions de conception donc de rebondir vers l'activité conceptuelle de conception. Le modèle de l'interaction mixte présenté au Chapitre 4 vise à permettre cette activité conceptuelle. Nous souhaitons permettre l'utilisation effective du modèle d'interaction mixte présenté au Chapitre 4 en ayant un outil de prototypage permettant de naviguer facilement entre les solutions conceptuelles modélisées et leurs matérialisations. Avoir un outil de prototypage qui opérationnalise notre modèle est une condition nécessaire à l'adoption de notre modèle d'interaction [Beaudoin-Lafon, 2004]. De plus, nous voyons dans cet objectif d'opérationnalisation du modèle une opportunité de répondre aux objectifs précédents de notre outil, grâce à la modularité à un niveau d'abstraction correspondant à l'objet mixte, intermédiaire entre dispositifs physiques (ressources matérielles) et applications.

Nos objectifs définis, nous présentons maintenant les outils de prototypage existants qui pourraient répondre à ces objectifs.

2. Tour d'horizon des outils de prototypage

Nous présentons ici les outils de prototypages existants. Nous structurons notre présentation suivant leur nature, matérielle ou logicielle. Lorsqu'un outil propose à la fois une contribution matérielle et logicielle, nous le présentons selon sa plus forte contribution. De plus, au-delà de la nature de l'outil, il conviendra au long de cette présentation de se poser la question suivante : dans quelle mesure ces outils sont-ils capables de répondre à nos objectifs ?

2.1. Outils matériels

Nous trouvons des outils matériels[37] pour l'aide au prototypage des interfaces mixtes. En effet, comme souligné dans le Chapitre 2 page 31 et dans [Myers et al., 2000], les ressources matérielles pour prototyper les interfaces mixtes sortent des standards clavier, écran et souris. Par conséquent, de tels outils sont nécessaires.
Parmi ces outils matériels, nous distinguons :

- Des outils purement matériels, utilisant des protocoles de communication avec la machine largement répandus,
- Des outils matériels utilisant des protocoles de communication dédiés et donc fournissant une interface de développement (API).

[37] hardware

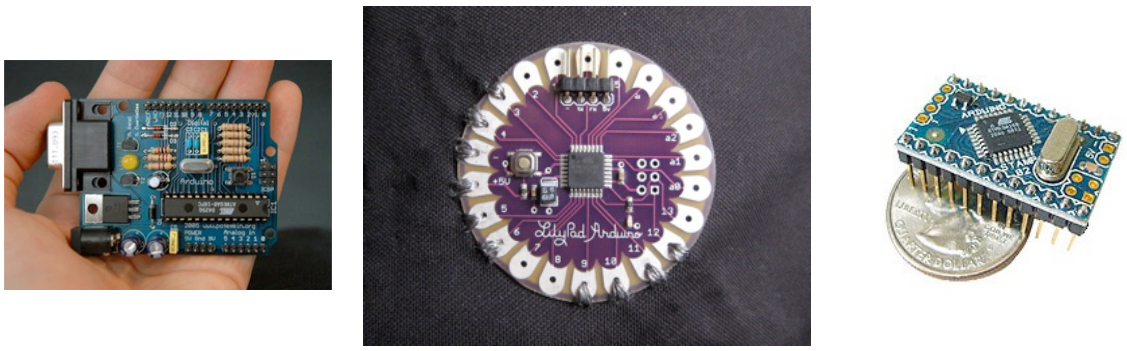
2.1.1. Capteurs et Effecteurs

Il existe des gammes de capteurs et effecteurs, disponibles sur le marché, qui utilisent différentes connectiques et protocoles variés. Parmi ceux-là, nous citons les dispositifs Interface-Z (http://www.interface-z.com/), I-CubeX (http://infusionsystems.com/), et La Kitchen (http://www.la-kitchen.fr/kitchenlab/kitchenlab.html, aujourd'hui fermée).
Selon les protocoles utilisés (MIDI, OSC, UDP, etc.), ils nécessitent une ou plusieurs interfaces afin de communiquer avec l'ordinateur, comment des interfaces MIDI, les interfaces Kroonde et Toaster de La Kitchen, les interfaces d'Interface-Z, ou les WiSe Box, EtherSense, AtoMIC Pro, Eobody émanants de l'IRCAM (http://recherche.ircam.fr/equipes/temps-reel/movement/flety/).
Ces dispositifs sont souvent utilisés avec des logiciels de programmation graphique tels que Max/MSP, payant (http://www.cycling74.com/products/max5), ou PureData, libre (http://puredata.info/). Mais ces dispositifs matériels peuvent être tout aussi bien utilisés sans ce type de logiciels. Ce matériel est très largement utilisé par les artistes des nouveaux médias, musiciens ou plasticiens, mais tend aujourd'hui à être remplacé par Arduino, que nous présentons maintenant.

2.1.2. Arduino

Arduino (http://www.arduino.cc/) est une plateforme de prototypage électronique libre, fournissant du matériel et du logiciel pour le programmer. Arduino a été pensé pour être facile à utiliser et flexible (adaptable par ses utilisateurs). Les utilisateurs ciblés sont les artistes, les designers, les bricoleurs amateurs voulant créer des objets et environnements interactifs.

Arduino propose à la vente une dizaine d'interfaces électroniques. La Figure 154 en présente trois : l'interface USB de base (a), l'interface Lily Pad, conçue pour pouvoir être cousue dans le tissu, et l'interface Arduino mini, la plus petite. D'autres interfaces existent, par exemple une technologie sans fil avec une carte Bluetooth. Les cartes elles-mêmes sont programmables, permettant à l'interaction d'être programmée au niveau matériel ou logiciel. Sur ces interfaces, branchées à l'ordinateur ou non, peuvent être branché plusieurs types de capteurs et d'effecteurs.



(**a**) (**b**) (**c**)

Figure 154 : Interfaces matérielles Arduino : (a) l'interface USB de base (Duemilanove), (b) Lily Pad cousue dans du tissu, et (c) l'Arduino mini. Illustrations extraites de http://www.arduino.cc/.

La partie logicielle d'Arduino consiste en un environnement de développement et des bibliothèques. Le logiciel est disponible pour Windows, Linux, et Mac OS X. Le langage de programmation Arduino est basé sur Wiring (http://wiring.org.co/) et sur C/C++. Afin de prototyper l'interaction, ce langage permet de programmer le microcontrôleur présent sur la carte.
L'environnement de développement est basé sur Processing (http://processing.org/), un langage Java simplifié utilisé par la communauté des arts et arts appliqués. Arduino peut également être facilement utilisé depuis Processing. Il est intéressant de noter que les choix qui ont été faits par Arduino ne sont pas ceux d'un environnement graphique de programmation, au contraire de Max/MSP ou PureData, malgré les utilisateurs visés (artistes, designers, amateurs, etc.).

2.1.3. Phidgets

Les Phidgets (« *Physical Widgets* », http://www.phidgets.com/) résolvent un problème technologique : il est difficile aux développeurs de logiciels de prototyper des applications qui utilisent du matériel non conventionnel, car ils n'ont souvent pas les connaissances et l'expertise pour et/ou les ressources (Documentations, APIs, etc.) nécessaires. De plus, si les ressources sont là, elles ne sont pas forcément adaptées à l'utilisation que les concepteurs veulent en faire (niveau d'abstraction trop bas ou trop haut par exemple, dépendance à un domaine d'application différent, etc.). L'effort nécessaire pour prototyper ce type d'interface est donc très grand.

Les Phidgets fournissent un ensemble de capteurs et effecteurs, qui peuvent être raccordés à l'ordinateur via des interfaces USB. La Figure 155 présente un exemple de kit proposé par Phidgets Inc. Ces dispositifs sont fournis avec une API et autorisent de nombreux langages (.NET, Visual Basic, VBA Microsoft Access et Excel, Java, C, C++, Python, etc.). Les Phidgets proposent une interface optionnelle qui affiche et permet de contrôler l'état des dispositifs. De plus, il est possible grâce à l'API fournie de prendre en compte leur apparition et disparition au sein de l'interface.



Figure 155 : Exemple de kit de Phidgets pour débuter : (de haut en bas et de gauche à droite) un capteur de toucher, une glissière, un capteur de lumière, un potentiomètre, un capteur de force, l'interface USB sur laquelle brancher les capteurs et effecteurs, 8 interrupteurs, et des LEDs. Illustration extraite de http://www.phidgets.com.

Les phidgets ont évolué de l'état de prototype de recherche [Greenberg, Fitchett, 2001] à un produit commercial (http://www.phidgets.com). Ils ont de plus été étendus pour les interfaces distribuées dans [Marquardt, Greenberg, 2007]. Les « Shared Phidgets » ou « Phidgets partagés » permettent de prototyper plus facilement des interfaces dont les éléments sont distribués sur des ordinateurs différents et sur le réseau.

2.1.4. Switcharoo

La problématique résolue par [Avrahami, Hudson, 2002] est celle de la combinaison de l'apparence et de l'interaction pour le prototypage à basse résolution. Comme ces dimensions s'influencent, les auteurs de [Avrahami, Hudson, 2002] proposent un outil pour pouvoir prototyper la forme *et* l'interaction dans les premiers prototypes.

L'outil proposé, nommé *Switcharoo*, consiste en un ensemble de dispositifs matériels sans fil : des boutons, des potentiomètres linéaires, et des joysticks à cinq directions. Ils sont construits à partir de marqueurs RFID à interrupteur, et peuvent ainsi communiquer sans être alimentés en énergie : ils sont donc petits. C'est un lecteur RFID connecté à l'ordinateur qui est alimenté et permet la communication avec ce dernier. Ces dispositifs peuvent ainsi être connectés à Director® (http://www.adobe.com/products/director/). Comme Switcharoo est pensé pour le prototypage, les dispositifs peuvent être facilement attachés et détachés d'un prototype physique explorant l'apparence de l'objet, grâce à des épingles (Figure 156).

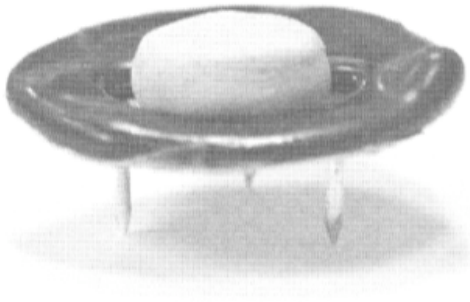


Figure 156 : Bouton de Switcharoo à épingler sur un prototype grâce à ses trois épingles. Illustration extraite de [Avrahami, Hudson, 2002].

La partie logicielle de Switcharoo consiste en une interface utilisateur pour le designer et une interface logicielle avec les capteurs RFID :

- L'interface pour le designer est une extension de Director®, autrefois Macromedia®, aujourd'hui Adobe®. Le designer doit alors prototyper l'interaction grâce à des évènements intégrés à Director®. Switcharoo ne change ainsi pas la pratique des designers avec Director®.
- L'interface avec les capteurs est basée sur java (http://java.sun.com/). Le logiciel de Switcharoo peut fonctionner avec Windows ou MacOS.

2.1.5. Calder

Calder [Lee et al., 2004] est une boîte à outils matérielle, composée de dispositifs d'entrée et de sortie, avec et/ou sans fil. La boîte à outils peut être utilisée via un environnement de prototypage d'interface laissé au choix du concepteur : les auteurs citent le langage de programmation java et Director®.
L'interface entre le matériel et le logiciel se fait grâce à un mécanisme de substitution : les détails d'implémentation à bas niveau sont regroupés et peuvent être accédés via des fonctions en C, que java (http://java.sun.com/) ou Director® peuvent exploiter. Ainsi les dispositifs peuvent générer des événements et réagir aux évènements.

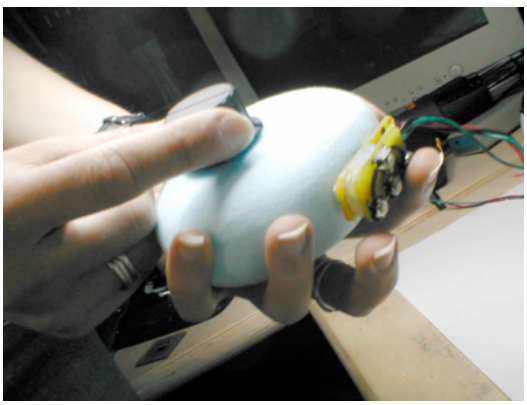


Figure 157 : Prototype fait de mousse pour explorer l'apparence et équipé de dispositifs de Calder pour explorer l'interaction. Illustration extraite de [Lee et al., 2004].

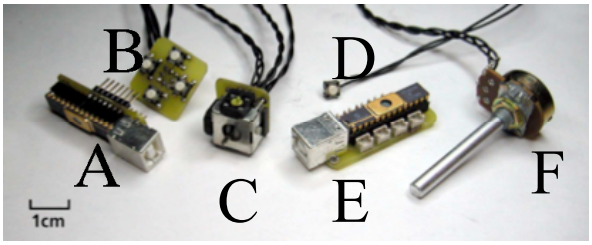


Figure 158: Dispositifs matériels filaires de Calder : (A) carte d'entrée/sortie, (B) ensemble de 4 boutons, (C) manette analogique, (D) simple bouton, (E) composant d'entrées numériques/analogiques, (F) potentiomètre rotatif. Illustration adaptée de [Lee et al., 2004].

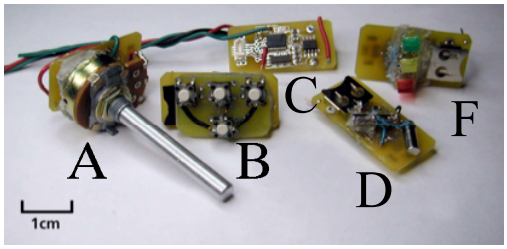


Figure 159: Dispositifs matériels sans fils de Calder : (A) potentiomètre rotatif, (B) ensemble de 4 boutons, (C) transmetteur sans fil, (D) capteur d'inclinaison à deux axes, (E) ensemble de trois LEDs. Illustration adaptée de [Lee et al., 2004].

À la différence de Switcharoo, Calder fait des concessions sur la taille et le sans-fil, afin d'avoir plus de possibilités d'interaction. Switcharoo est limité, avec les interrupteurs comme composants élémentaires, alors que Calder fournit d'autres possibilités d'interaction, bien qu'il ait des composants plus gros et filaires. Parmi les dispositifs sans fils (Figure 159), le transmetteur (C) est un dispositif qui fait l'interface entre l'ordinateur et les dispositifs sans fil : il communique donc à la fois avec et sans fil.

2.1.6. BOXES

La problème noté par [Hudson, Mankoff, 2006] est le même que pour les outils précédents : le prototypage combiné de la forme et de l'interactivité. BOXES (Building Objects for eXploring Executable Sketches) est le descendant de Switcharoo et Calder.

Cet outil de prototypage consiste en une partie matérielle associée à une partie logicielle. Le concepteur peut brancher un ou des fils sur les capteurs de toucher d'une interface matérielle connectée via une connexion USB à l'ordinateur. Le concepteur peut ensuite, via une interface logicielle, affecter à chaque entrée filaire de la carte (interface matérielle) une entrée logicielle sous la forme d'une simulation d'événement clavier ou souris à un endroit particulier de l'écran. Par exemple à la Figure 160, une punaise fixée au prototype (« Thumbtack 0 », Figure 160 à droite) est associée au simple clique sur le bouton « lecture » de RealOne Player : lorsque l'utilisateur presse la punaise, la lecture du morceau de musique est lancée. Dans l'interface sur l'écran de la Figure 160, nous trouvons :

- à gauche, la sélection de la punaise,
- au milieu, les actions clavier/souris qui sont simulées lorsque la punaise est touchée,
- à droite, une copie d'écran du lieu sur l'écran où ces actions clavier/souris seront simulées.

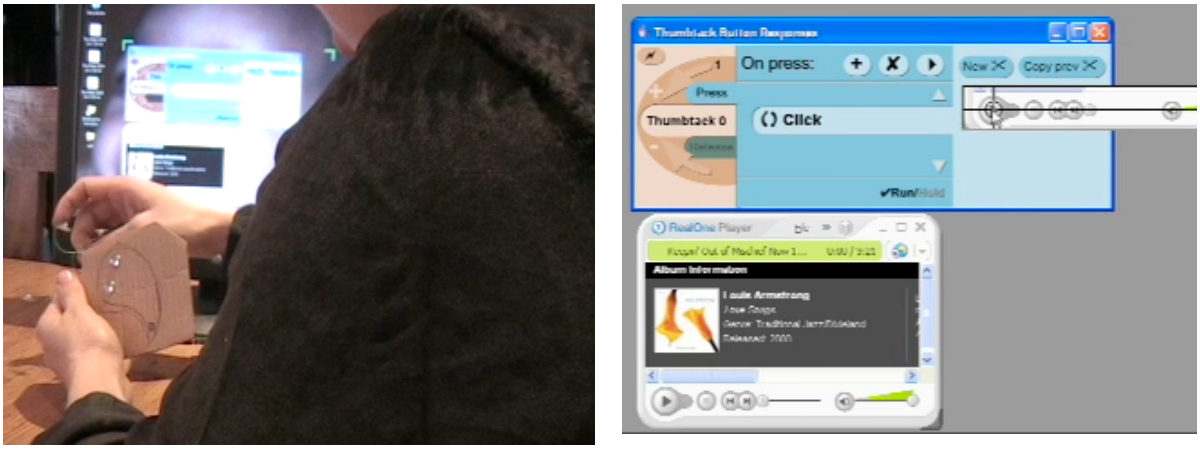


Figure 160 : BOXES, un outil pour combiner le prototypage à basse résolution de l'apparence et d'interaction. Illustration extraite de la vidéo de [Hudson, Mankoff, 2006].

Les concepteurs peuvent explorer l'apparence de leur design avec des matériaux comme le carton, le papier, le scotch, et avec des objets métalliques ordinaires, comme des punaises, des trombones, ou encore du papier aluminium. Ces parties métalliques peuvent être connectées par un fil à l'interface matérielle de BOXES et participer à l'interaction.

Les différences fondamentales avec les outils précédents sont que :

- BOXES permet d'utiliser des matériaux ordinaires (punaises, papier aluminium) comme dispositifs d'interaction. Les possibilités d'interaction sont donc plus limitées qu'avec Calder ou Switcharoo.
- Tous les dispositifs de BOXES (punaises, papier aluminium, etc.) sont filaires, contrairement aux besoins identifiés dans [Avrahami, Hudson, 2002], confirmant les concessions effectuées depuis Switcharoo.
- BOXES permet de faire tout ce qu'une souris et un clavier peuvent faire et peut donc fonctionner avec des applications faites avec Director®, Flash® ou Java, mais aussi des applications commerciales faites par un tiers. Les auteurs se sont donc dirigés vers une interface logicielle plus indépendante de l'application, tout en restant utilisable pour des non-informaticiens. La limite inhérente à ce choix est que BOXES ne permet de prototyper que des interfaces physiques pour des applications déjà existantes, ce qui n'est pas toujours le cas en pratique.

Nous venons de présenter des outils matériels ou hybrides. Ces outils ont un niveau d'abstraction trop bas (proche du matériel) pour répondre totalement à nos objectifs. En particulier, ils n'explicitent pas la notion d'objet mixte, notion centrale à notre modèle d'interaction mixte. Néanmoins, leur contribution est essentielle et se doit d'être capitalisée au sein d'outils de plus haut niveau d'abstraction.

2.2. Outils logiciels

Après avoir présenté des outils matériels pour le prototypage, nous nous intéressons aux outils logiciels.

2.2.1. ARToolKit

L'ARToolKit[38] (http://www.hitl.washington.edu/artoolkit/) est une bibliothèque logicielle pour prototyper des applications en réalité augmentée. Elle permet de superposer des images numériques sur le champ de vision de l'utilisateur. Par exemple à la Figure 161, l'image en trois dimensions d'un

[38] Boîte à outils pour la réalité augmentée

personnage (numérique) est affichée comme s'il se tenait sur le plateau (physique) tenu par l'utilisatrice. Elle peut voir ce personnage à travers les lunettes semi-transparentes qu'elle porte.

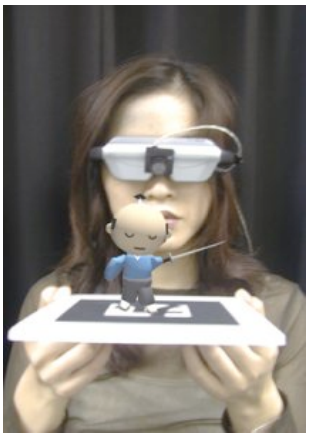


Figure 161 : Exemple d'application construite avec l'ARTolKit.

L'ARToolKit permet de résoudre les problèmes technologiques liés à la vision par ordinateur. Beaucoup de développeurs d'interfaces homme-machine n'ont pas les compétences et l'expérience requises pour développer ce type d'applications. L'ARToolKit propose une boîte à outils utilisable par ces développeurs.

L'ARToolKit permet de calculer en temps réel la position et l'orientation de la caméra relativement à un ou des marqueurs, afin de suivre leurs positions et d'afficher correctement l'image numérique. Ces marqueurs, qui peuvent être réalisés par le concepteur, doivent comporter un motif dans un carré noir. C'est une limitation que des extensions de l'ARToolKit ont proposé de résoudre : avec l'AiRToolKit (Augmented interactive Reality toolkit, http://sketchblog.ecal.ch/variable_environment/) par exemple, l'utilisateur peut dessiner ses propres marqueurs sur un papier pré-quadrillé (Figure 162 en haut à gauche) en lui donnant des formes diverses. Il les prend ensuite en photo (Figure 162 en haut au centre) pour les lier à un élément numérique, comme une étoile en 3D (Figure 162 en bas au centre, sur l'écran), ou à une fonction, comme allumer la lumière (Figure 162 en bas au centre). De nombreuses formes sont possibles (Figure 162 en bas à gauche et à droite).

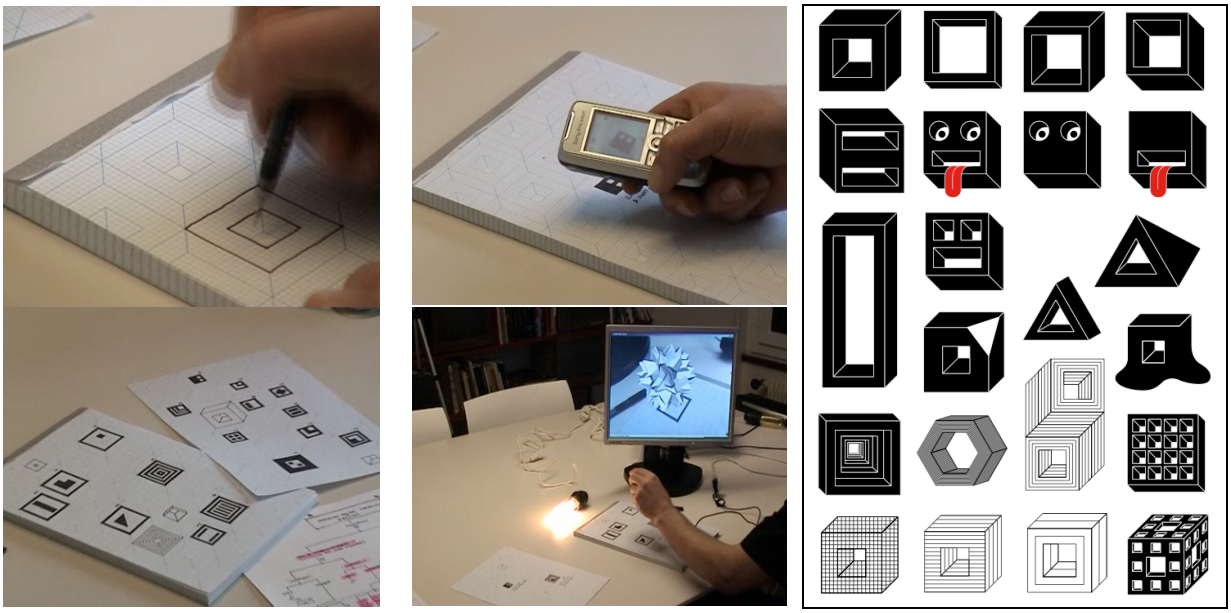


Figure 162 : AiRtoolkit : Extension de l'ARToolKit pour reconnaître de nouveaux types de marqueurs (à droite). L'utilisateur peut dessiner ses propres marqueurs sur un papier pré-quadrillé (en haut à gauche) et les prendre en photo (en haut au centre) pour les lier à un élément numérique, comme une étoile en 3D, ou à une fonction, comme allumer la lumière (en bas au centre). De nombreuses formes sont possible (en bas à gauche). Illustrations extraites de http://sketchblog.ecal.ch/variable_environment/.

L'ARToolKit permet également de calibrer la caméra facilement. La boîte à outils est mutiplateforme, et les sources sont disponibles (http://www.hitl.washington.edu/artoolkit/download/). De plus, depuis son apparition, elle a été portée sur d'autres plateformes, tel l'iPhone récemment (http://www.artoolworks.com/ARToolKit_iPhone.html).

2.2.2. Papier-Mâché

Papier-Mâché [Klemmer et al., 2004] est une boîte à outils pour aider le prototypage des interfaces tangibles utilisant la vision par ordinateur, les codes barres et la technologie RFID (« Radio Frequency IDentification »). Comme l'ARToolKit, la motivation pour Papier-Mâché est la résolution d'une difficulté technologique. Pour pouvoir jongler facilement entre ces différentes technologies, Papier-Mâché a introduit une gestion des évènements d'interaction faisant abstraction des technologies utilisées (vision par ordinateur, codes barres ou RFID).

Papier-Mâché représente les objets physiques avec la notion de **Phobs**. En entrée du système, le logiciel réalise :

1. l'acquisition des données du capteur (caméra, lecteur RFID, etc.),
2. l'interprétation des données,
3. la génération des Phob(s).

Le développeur doit choisir le type de dispositif d'entrée (caméra, lecteur de codes barres ou RFID), mais n'a pas besoin de traiter la gestion des connexions/déconnexions des dispositifs et des évènements qu'ils envoient. C'est Papier-Mâché qui génère les évènements liés aux apparitions, disparitions et mises à jour des dispositifs, à travers des évènements identiques quelle que soit la technologie utilisée (même si le type d'information véhiculé par ces évènements est évidemment différent selon la technologie).
Pour la vision par ordinateur, Papier-Mâché crée un **VisionPhob** (Figure 163), qui lui-même sera responsable de la création d'évènements de type **phobAdded**, **phobUpdated**, **phobRemoved**. Dans la Figure 164, nous voyons en haut à gauche des Phobs générés par la vision par ordinateur.

PhobProducer prod = new VisionPhobProducer(new CameraImageSource()) ;

Figure 163 : Ligne de code écrite avec Papier-Mâché pour créé un VisionPhob.

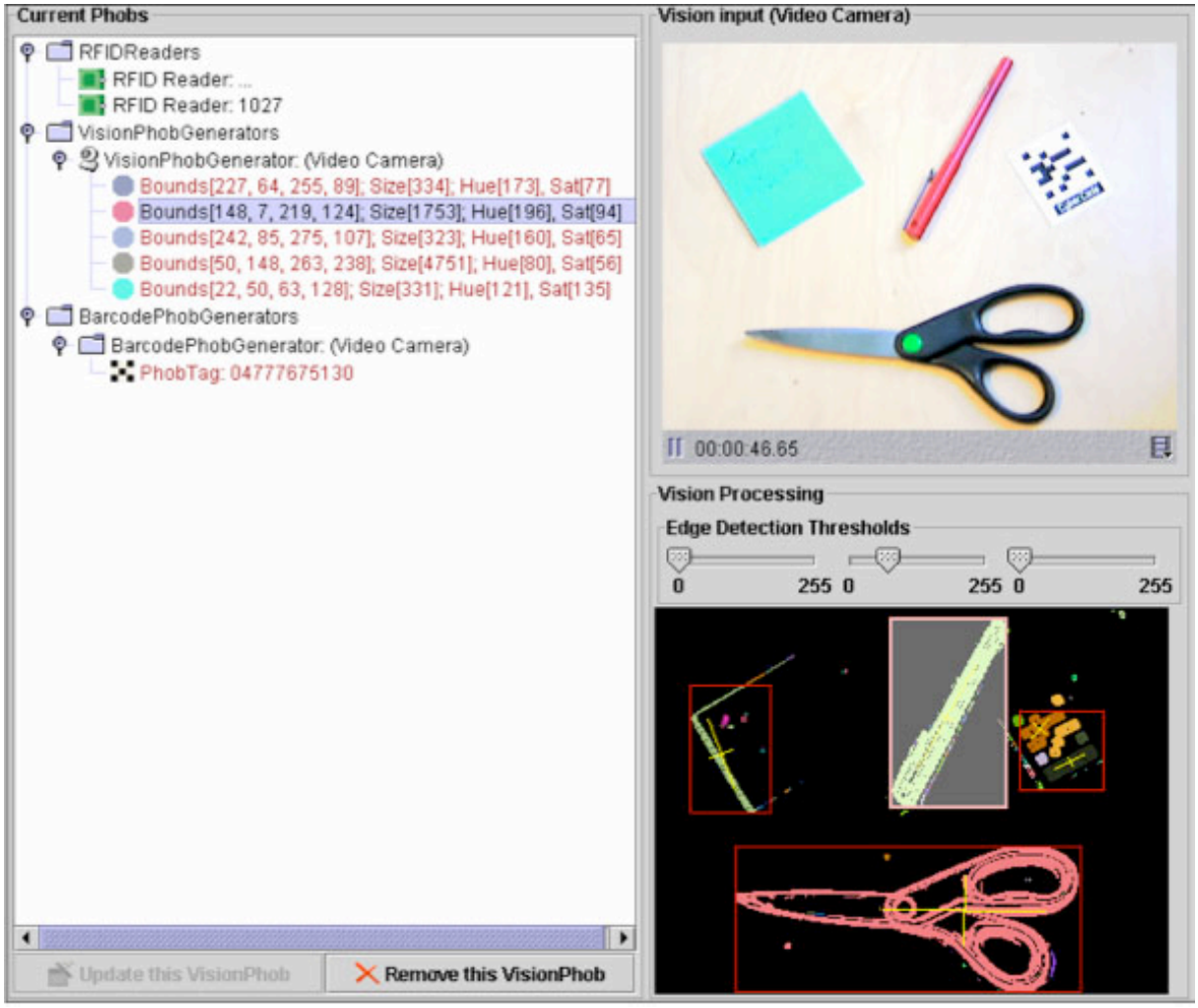


Figure 164 : Interface pour le développeur travaillant avec Papier-Mâché (partie de gauche). Illustration extraite de [Klemmer et al., 2004].

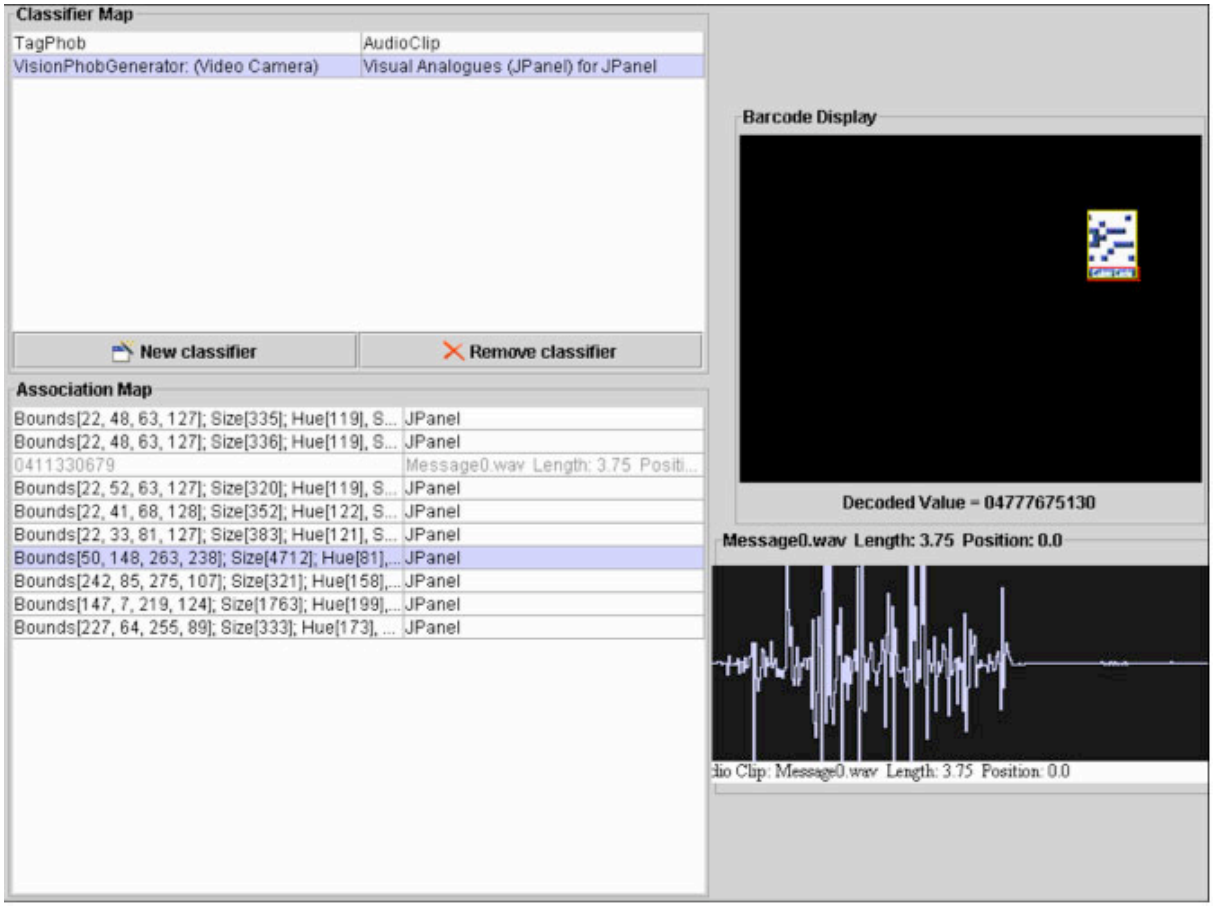


Figure 165 : Interface pour le développeur travaillant avec Papier-Mâché (partie de droite). Illustration extraite de [Klemmer et al., 2004].

Une fois la gestion des dispositifs d'entrée spécifiée, le développeur peut associer les Phobs avec des éléments numériques. Il doit pour cela implémenter une **AssociationFactory**. Celle-ci attend (« écoute ») les évènements entrants, pour mettre à jour l'objet numérique associé. Cet objet numérique associé peut être de l'information (appelé « nom »), ou une commande, un traitement (appelé « verbe ») [Klemmer et al., 2004]. La Figure 166 montre comment on associe des Phobs avec des éléments numériques.

```
AssociationFactory factory = new VisualAnalogueFactory(
        new PMacheWindow(gen, CALIBRATE),
        JPanel.class) ;
AssociationMap assocMap = new AssociationMap(factory) ;
gen.addPhobListener(assocMap) ;
```

Figure 166 : Ligne de code écrite avec Papier-Mâché pour associer les Phobs avec des éléments numériques.

Papier-Mâché fournit au développeur une interface (Figure 164 et Figure 165) pour observer et contrôler le déroulement de l'interaction. Dans la partie gauche de cette interface (Figure 164) est affiché un arbre de tous les dispositifs et les Phobs qu'elles détectent. Est également affiché l'image captée par la caméra, l'image après traitement, ainsi que les associations (Figure 165, partie droite). Cette interface peut servir aussi de Magicien d'Oz : le développeur peut ajouter ou supprimer des évènements.

Au regard de nos objectifs, cette boîte à outils apporte de nouvelles caractéristiques par rapport aux précédentes : l'utilisation quasi-transparente de différentes technologies et l'association entre physique et numérique. Pourtant, elle est encore trop loin du cadre défini par le modèle

d'interaction mixte. En effet, elle est limitée à trois types de dispositifs, et ne respecte pas le niveau d'abstraction de l'objet mixte, puisqu'un Phob peut être associé à une commande (niveau interaction) ou à de l'information (niveau objet).

2.2.3. IStuff & IStuff mobile

Istuff et IStuff mobile sont des boîtes à outils apparentées, que nous présentons l'une après l'autre, pour le prototypage d'interfaces distribuées dans une pièce ou sur un téléphone mobile.

2.2.3.1. IStuff

IStuff [Ballagas et al., 2003] est une boîte à outils qui permet d'intégrer des dispositifs physiques multiples disponibles à un instant donné dans une pièce pour interagir avec plusieurs utilisateurs co-localisés, comme dans l'iRoom de Stanford University (Figure 167). La pièce où se trouvent les utilisateurs possède des ressources multiples et hétérogènes : elle héberge par exemple un système ayant plusieurs écrans de différentes tailles et intègre différentes technologies sans fil. Les utilisateurs peuvent interagir avec plusieurs applications, de façon concurrente.



Figure 167 : iRoom de Stanford University.

Dans une pièce comme l'iRoom où interagissent plusieurs utilisateurs, IStuff est une boîte à outils qui permet d'intégrer des dispositifs physiques multiples disponibles à un instant donné. Les **dispositifs iStuff** (Figure 168) ont deux parties (Figure 169) :

1. Le dispositif sans fil en lui-même,
2. La partie logicielle du dispositif qui reçoit les données, s'exécutant sur une machine située dans la pièce, servant de « Proxy ».

Ainsi les ressources informatiques et matérielles nécessaires pour le dispositif sans fil en lui-même sont minimisées. C'est une motivation que nous trouvons aussi dans Switcharoo [Avrahami, Hudson, 2002] (section 2.1.4). Le protocole de communication sans fil (Fréquence Radio, Infra rouge, Bluetooth, etc.) est encapsulé dans la partie logicielle du dispositif sur le Proxy, afin que la technologie ne soit pas un frein au développement. La Figure 168 montre des dispositifs iStuff développés, auxquels s'ajoutent les commandes vocales, et le retour sonore via des haut-parleurs sans fil. Par exemple, l'iDog envoie un évènement « bouton pressé » (comme la souris) lorsqu'il est retourné par l'utilisateur.



Figure 168 : Dispositifs iStuff. Illustration extraite de [Ballagas et al., 2003].

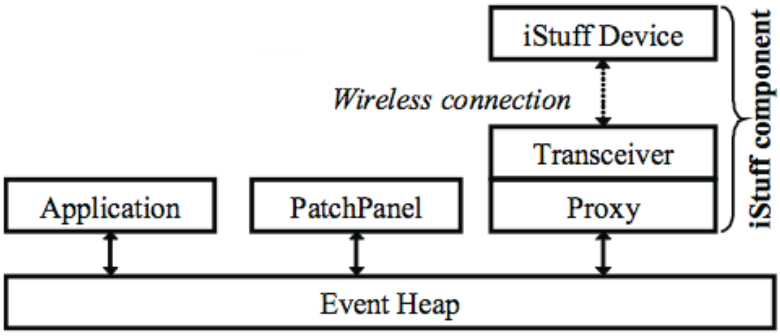


Figure 169 : Architecture d'iStuff. Illustration extraite de [Ballagas et al., 2003].

La communication entre les dispositifs iStuff et les applications se fait par événements sur le réseau. Un évènement est un message contenant un type et optionnellement des pairs (clé, valeur). Ensuite, la **pile d'événements** (**Event Heap**) (Figure 169) est un processus qui :

- reçoit les évènements d'un producteur (dispositif iStuff ou application),
- les redistribue à ses destinataires (applications ou dispositifs iStuff) depuis le serveur où il s'exécute. Les destinataires peuvent spécifier le type d'évènements recherchés et d'autres paramètres pour ne récupérer que les évènements qui les intéressent.

Cette communication est basée sur iRos [Johanson et al., 2002.] qui sert de middleware commun entre les dispositifs d'interaction et les applications. IRos est basé sur TCP et Java, ce qui permet au système d'être multiplateforme. Il offre une communication asynchrone via la pile d'évènements (Event Heap).

Pour interfacer ces dispositifs hétérogènes avec des applications multiples, il est nécessaire de pouvoir faire correspondre la sortie des dispositifs avec l'entrée des applications. Pour cela, les auteurs proposent le **Patch Panel** (Figure 169) qui permet de configurer dynamiquement (c'est-à-dire pendant l'exécution de l'application) la correspondance entre dispositifs et applications, via une interface Web. Ainsi les utilisateurs finaux peuvent configurer leurs interfaces selon les ressources matérielles dont ils disposent. Le Patch Panel est un client de la pile d'évènements. Il y écoute les évènements pour lesquels il a été configuré et les transforme avant de les remettre dans la pile d'évènements. Les évènements du côté application et du côté dispositif peuvent alors être indépendants. Par exemple, une application de dessin prend comme entrée un événement « Déplacer Pinceau » (indépendant du dispositif) et les dispositifs proposent des événements comme « Souris Déplacée » ou « Glissière Modifiée ». C'est le Patch Panel qui permet d'associer les sorties des dispositifs iStuff avec les entrées des applications. Toutes les boîtes à outils qui cherchent à prendre en compte plusieurs dispositifs se trouvent confrontées à la question de l'interopérabilité, et IStuff propose une réponse souple.

Une application, pour être utilisée avec iStuff, doit être adaptée afin de prendre en compte la pile d'évènements (Event Heap). Pour ajouter un dispositif qui puisse être utilisable dans iStuff, il faut écrire le logiciel d'interfaçage avec la pile d'évènements (Event Heap) utilisée par iStuff. Chaque dispositif iStuff possède son type d'évènement.

La flexibilité offerte par IStuff permet de prototyper l'interaction entre de multiples dispositifs et applications. Néanmoins, IStuff ne permet pas d'ajuster les dispositifs de façon fine. Ces dispositifs sont rigides et ne constituent pas le centre d'intérêt de la conception. Pourtant, la contribution d'IStuff est importante et il serait pertinent de le considérer pour un plus bas niveau d'abstraction. En cela, nous situons iStuff à un plus haut niveau d'abstraction que les précédents outils présentés.

2.2.3.2. IStuff Mobile

IStuff mobile [Ballagas et al., 2007] est une boîte à outils dédiée au prototypage d'interaction sur téléphones mobiles standards (équipés du système d'exploitation Symbian Series 60) avec du matériel externe au téléphone. IStuff mobile est donc une boîte à outils résolvant un problème technologique. Pour cela, IStuff mobile fournit les éléments (violets) de la Figure 170 :

- Un Proxy (Figure 170 en haut à gauche « Smart-Its Proxy ») permet de :
  - Configurer le capteur,
  - Recevoir les données du capteur,
  - Poster les évènements dans la pile d'évènements d'iStuff.
- Une partie logicielle pour le téléphone mobile s'exécute en tâche de fond (Figure 170, B) et permet de prototyper pour n'importe quelle application qui s'exécute en premier plan (Figure 170, D). Pour cela, les deux applications communiquent via des évènements système, comme des évènements clavier. L'application en tâche de fond communique avec la pile d'évènements d'IStuff à l'exécution.
- Un éditeur graphique basé sur Quartz Composer (http://developer.apple.com/graphicsimaging/quartz/quartzcomposer.html) permet de spécifier l'interaction (Figure 170, C). Les auteurs ont ajouté à la bibliothèque existante dans Quartz Composer des composants pour les Proxys et pour des traitements de données qui étaient utilisés dans les applications mobiles et encore non traités dans Quartz Composer.

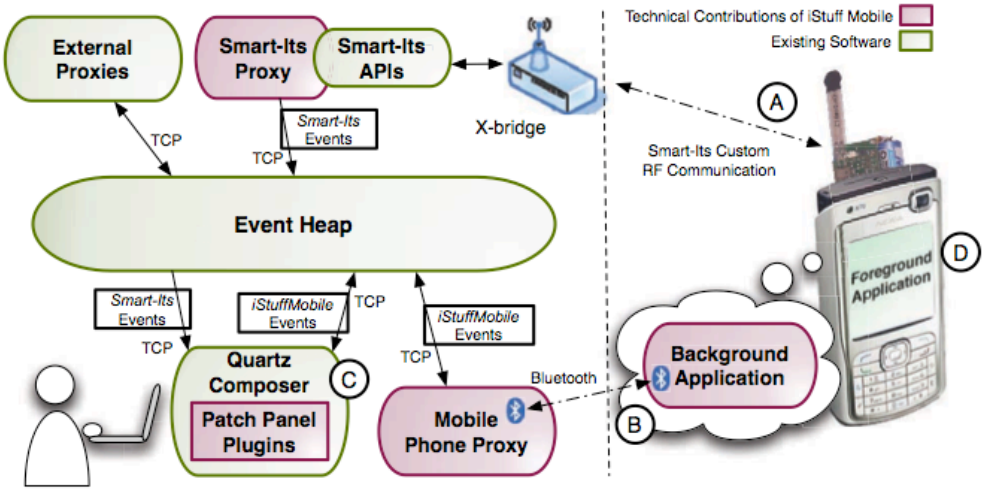


Figure 170 : Architecture d'iStuff mobile. Illustration extraite de [Ballagas et al., 2007].

IStuff mobile, au contraire d'autres boîtes à outils comme les Phidgets (http://www.phidgets.com/), ne comporte qu'une partie logicielle. IStuff capitalise les progrès faits dans le prototypage du côté matériel et intègre des éléments matériels largement répandus. De plus, même du côté logiciel, IStuff mobile étend des outils existants comme iStuff, voire commercialisés comme Quartz Composer. IStuff fournit à iStuff mobile la possibilité de communiquer entre dispositifs et applications via le réseau, et de configurer cette communication pendant l'exécution grâce au Patch Panel. Quartz Composer, quant à lui, fournit à iStuff mobile un éditeur dynamique (pas de modes édition et exécution séparés : on peut éditer pendant l'exécution).

Même si le domaine d'application de l'outil (l'interaction sur téléphone) est limité à une plateforme et est trop restreint pour répondre à nos objectifs, nous retenons de cette approche la volonté de capitalisation d'outils existants.

2.2.4. Context Toolkit

La Context Toolkit [Salber et al., 1999] est une boîte à outils qui permet de construire une interaction sensible à l'information environnementale, telle que la position dans l'espace, l'identité d'objets ou de personnes, la température, l'ensoleillement, mais aussi l'environnement logiciel et matériel. Le défi relevé par la Context Toolkit est celui de la difficulté d'exploitation de ces informations pour l'interaction, due à l'hétérogénéité des ressources, qu'il faut parfois combiner, qui sont parfois distribuées sur le réseau, issues de matériels non conventionnels (autres que souris et clavier), et qui n'ont pas forcément le niveau d'abstraction requis pour l'application visée. De plus, contrairement aux interfaces conventionnelles, ces ressources ne sont pas fixes, mais peuvent changer au cours du temps, même pendant l'interaction.

Pour répondre à ces problèmes, les auteurs proposent des **context widgets**. Les context widgets encapsulent le dispositif matériel, parfois distribué, ainsi que la partie logicielle associée permettant de recueillir l'information qu'il délivre. Un context widget permet :

1. L'indépendance entre les ressources matérielles et l'application,
2. L'encapsulation des détails d'interaction et fournit à l'application l'information *nécessaire*, ni plus ni moins,
3. La réutilisation, l'adaptation et la combinaison entre elles ces briques logicielles génériques.

L'état d'un contexte widget est un ensemble d'attributs qu'une application peut interroger. Une application peut aussi souscrire aux fonctions de rappel[39] associées aux attributs, pour être automatiquement informée quand l'état d'un widget change.

Les context widgets sont implémentés avec des **générateurs**, des **interpréteurs** et des **serveurs**.

- Les générateurs encapsulent les capteurs matériels et la partie logicielle permettant de capturer l'information brute de ces capteurs. Pour implémenter un widget, il est possible d'utiliser un générateur ou plusieurs générateurs combinés. Changer de générateur ne change pas les attributs et fonctions de rappels d'un widget.
- Les interpréteurs permettent de transformer cette information brute vers un plus haut niveau d'abstraction. Ils permettent aussi la composition de widgets entre eux : dans ce cas, l'interpréteur analyse les informations émanant de deux widgets ou plus, et déduit une nouvelle information combinée. Il n'existe pourtant pas de composition temporelle des informations [Salber et al., 1999]. En revanche, une composition permettant de resoudre les ambiguïtés entre les informations issues de deux sources redondantes est proposée dans [Dey et al., 2002] grâce à un élément appelé **médiateur**.
- Les serveurs collectent, stockent et interprètent les informations venant d'autres widgets.

Ces trois types de composants logiciels peuvent être distribués sur le réseau, et communiquent via un mécanisme d'évènement basé sur les principes de souscription et d'interrogation. Un répertoire de composants est maintenu). Cette communication est implémentée en utilisant TCP/IP, HTTP et XML, qui sont largement répandus et multiplateformes.

Comme pour d'autres boîtes à outils, comme Papier-Mâché, cette boîte à outils permet l'utilisation quasi-transparente de différentes technologies. De plus le niveau d'abstraction est flexible. Pourtant, elle aussi est encore trop loin du cadre défini par le modèle d'interaction mixte : les modalités supportées sont très spécifiques.

[39] callbacks

2.2.5. d.tools

d.tools [Hartmann et al., 2006] (http://hci.stanford.edu/dtools/) est un outil de prototypage rapide des interfaces physiques, intégré dans un cycle itératif de conception comprenant la conception, les tests, et l'analyse des résultats expérimentaux, avant de reprendre la conception. La Figure 171 montre d.tools pendant son utilisation par des étudiants.

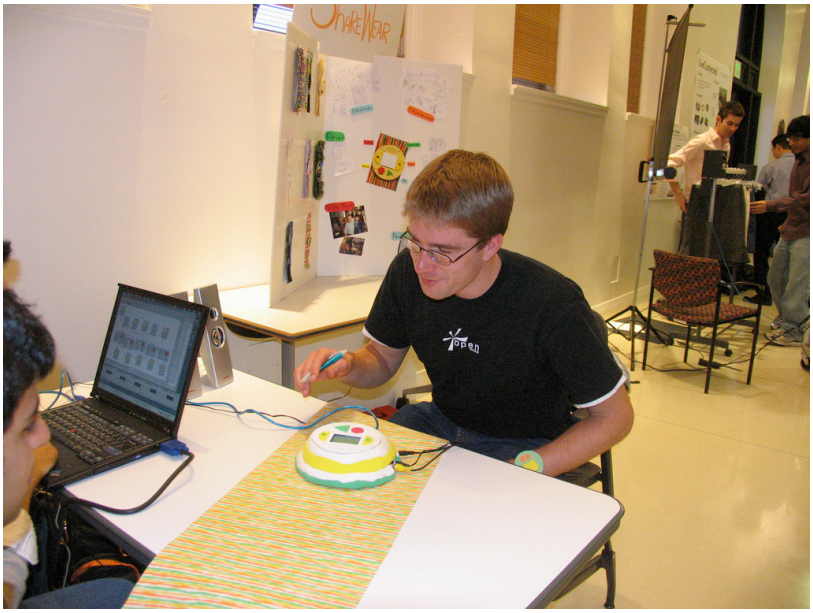


Figure 171 : d.tools pendant son utilisation. L'éditeur graphique est affiché sur l'écran de l'ordinateur, et le prototype sur la table est branché à l'ordinateur.

Le cycle itératif de conception avec d.tools s'organise en trois parties :

1. En mode conception, les concepteurs placent les dispositifs matériels intégrés à d.tools (Figure 172) sur le prototype et les connectent directement à l'ordinateur ou à la carte matérielle fournit par d.tools. Cette carte est branchée à l'ordinateur via USB. Par exemple, un dispositif peut être un mini-écran (en haut à droite de la Figure 172). Le concepteur peut ensuite arranger une image numérique des dispositifs pour représenter l'interface physique en glissant-déposant des composants élémentaires : à la Figure 173 à gauche (partie 1), une interface physique formée d'un écran dans sa partie supérieure et de cinq boutons dans sa partie inférieure est affichée. Si les dispositifs physiques ne sont pas connectés, le concepteur peut tout de même concevoir l'interaction, et simuler l'interface physique via l'éditeur de la Figure 173 à gauche (partie 1). d.tools fournit ensuite aux concepteurs une interface graphique basée sur un diagramme d'état (Figure 173, en haut à droite, partie 2). Les états y sont représentés grâce à l'image numérique de l'interface physique. L'état courant est entouré (de rouge) dans l'interface (Figure 173, au milieu à gauche, partie 2). Le diagramme d'état peut aussi être annoté/commenté et manipulé directement pour tester ce qui a été conçu. Les transitions représentées par des flèches relient entre eux les états. Ces transitions peuvent utiliser des ET/OU logiques. Des transitions automatiques peuvent être réalisées grâce à des *timers*[40]. Afin d'élargir les possibilités, les concepteurs peuvent aussi créer plusieurs diagrammes d'état fonctionnant en parallèle, ainsi qu'écrire du code java pour spécifier un état plus complexe dans l'éditeur (Figure 173, partie 3).

[40] minuteurs, compte à rebours

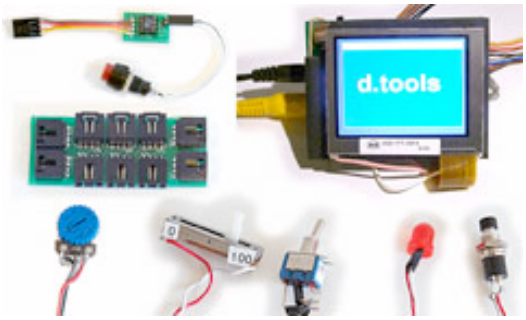


Figure 172 : Dispositifs matériels intégrés dans d.tools. Illustration extraite de http://hci.stanford.edu/dtools/.

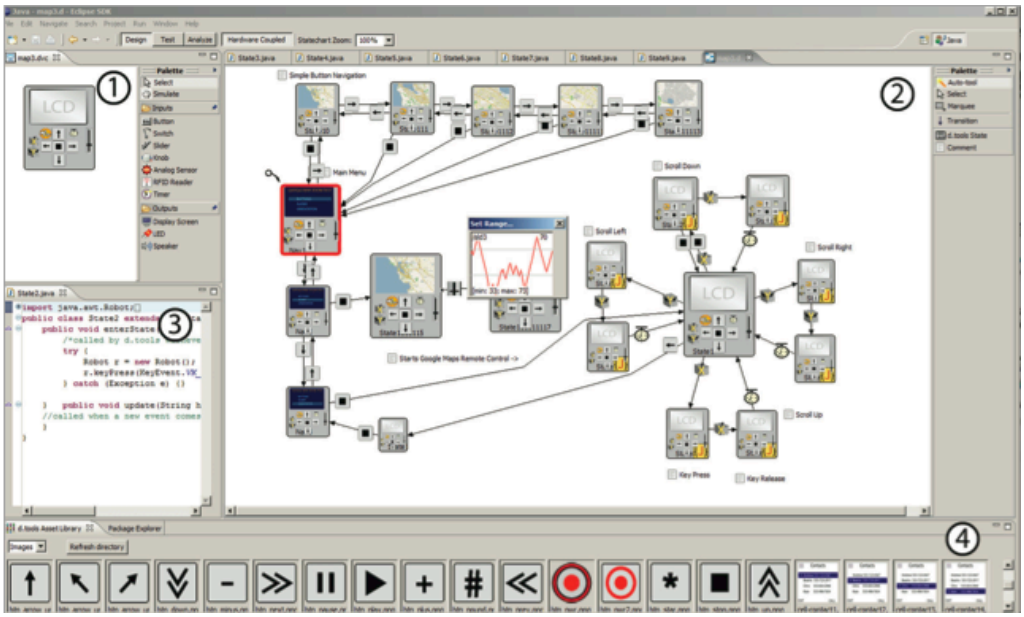


Figure 173 : Interface graphique de d.tools pour la conception. Illustration extraite de [Hartmann et al., 2006].

2. En mode test, le prototype peut être manipulé directement, et/ou son image logicielle (partie 1 de la Figure 173) peut servir de magicien d'Oz. d.tools réalise un enregistrement vidéo de l'interaction entre l'utilisateur et l'interface physique, ainsi qu'une liste de traces des évènements logiciels concernant l'interaction. Les traces logicielles sont utilisées pour structurer la vidéo. Ainsi un diagramme d'état pourra être directement mis en correspondance avec la ligne de temps de la vidéo enregistrée, de façon à accéder aux données par l'une ou l'autre représentation. L'interface physique est connectée à d.tools, et le mode test peut être lancé à n'importe quel moment. Pendant les tests, les concepteurs peuvent annoter/commenter l'enregistrement, avec deux types d'annotations : positive (pour un commentaire intéressant de la part de l'utilisateur, par exemple) et négative (pour un problème dans l'interface).
3. En mode analyse, l'interface graphique de d.tools permet d'accéder rapidement aux séquences vidéos pertinentes (Figure 174) et de comparer les différentes vidéos (Figure 175), grâce à l'enregistrement vidéo et les traces logicielles mis en correspondance. La ligne de temps de la vidéo fournit :
    a. les annotations (positives, négatives) faites en direct pendant le test par le concepteur,
    b. les états parcourus (Figure 174, partie 1), correspondant aux états de la partie 4,
    c. les évènements d'interaction (Figure 174, partie 3).

    Les concepteurs peuvent accéder aux données de l'évaluation via le diagramme d'état ou la ligne de temps de la vidéo. De plus, si le concepteur manipule directement l'interface physique pendant l'évaluation, d.tools met en valeur les évènements d'interaction similaires

sur la vidéo. La manipulation d'une vue des données d'évaluation entraîne la mise à jour des autres vues. Pour comparer plusieurs tests, les vidéos et les lignes de temps sont présentées en parallèle (Figure 175).

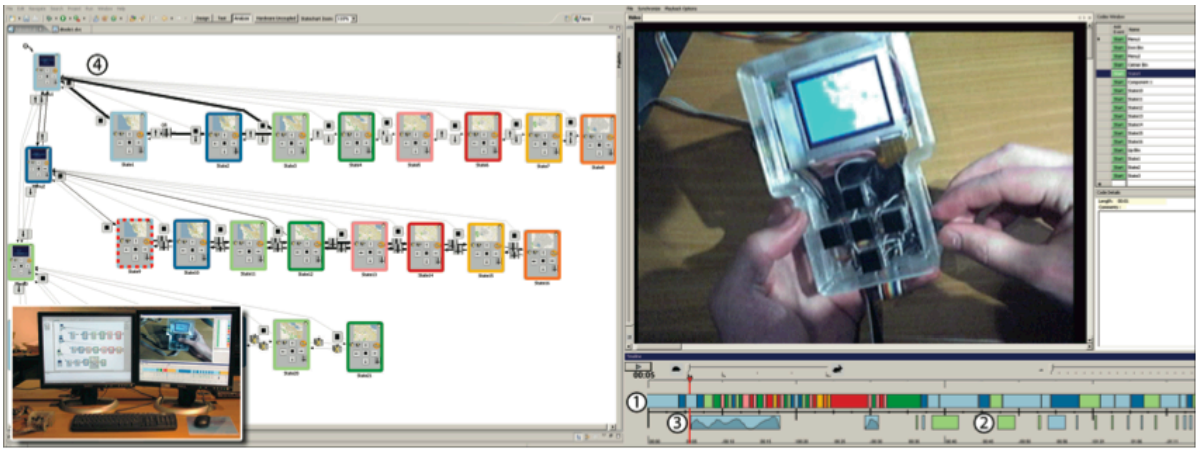


Figure 174 : Évaluation d'un prototype avec d.tools. Illustration extraite de [Hartmann et al., 2006].



Figure 175 : Interface graphique de d.tools pour comparer les données expérimentales de quatre évaluations. Illustration extraite de [Hartmann et al., 2006].

d.tools a été conçu pour être extensible. L'ensemble des dispositifs matériels (Figure 172) qui peuvent être utilisés est extensible, et ce à plusieurs niveaux.
La communication entre les dispositifs matériels et l'ordinateur se fait via le protocole OSC[41]. Ainsi pour ajouter des dispositifs matériels à ceux de base, il faut écrire une couche de code transformant leur protocole en OSC. Parmi ceux qui ont été ajoutés à d.tools, nous trouvons notamment les Phidgets (section 2.1.3) et Arduino (section 2.1.2). À un niveau d'abstraction plus bas, des électroniciens peuvent également développer des dispositifs pouvant être utilisés par la carte matérielle de d.tools.
L'extensibilité logicielle, quant à elle, consiste à proposer aux utilisateurs-concepteurs la possibilité d'écrire du code java s'exécutant lorsque le prototype est dans un état particulier.

Au regard de nos objectifs, la contribution de d.tools est orthogonale : la structure proposée par d.tools (diagramme d'états traduisant la trajectoire d'interaction) n'est pas celle de notre modèle d'interaction mixte (objets mixtes prenant part à l'interaction). Pourtant la contribution de d.tools et

[41] Open Sound Control

en particulier le couplage du prototypage avec l'évaluation, comme souligné dans [Myers et al., 2000], est une perspective intéressante.

2.2.6. Exemplar

Exemplar [Hartmann et al., 2007] est un outil qui permet de spécifier comment les données produites par un capteur vont être interprétées par le système. Pour cela, Exemplar permet au concepteur de l'interaction de :

1. faire une démonstration de l'utilisation du capteur. Ces données sont capturées, enregistrées et affichées graphiquement (Figure 176, *demonstrate*).
2. corriger ces données exemples sur l'interface, en manipulant le graphe des données captées en fonction du temps (Figure 176, *edit*),
3. vérifier que ce nouveau « modèle » corrigé correspond à l'interaction souhaité, en exécutant une nouvelle fois une démonstration (Figure 176, *review*).

Une fois le concepteur satisfait, cette interprétation des données peut être utilisée par un autre outil de prototypage, comme d.tools par exemple. Pour être intégré à un autre outil de prototypage, Exemplar peut exporter les évènements des capteurs en évènements clavier et souris.

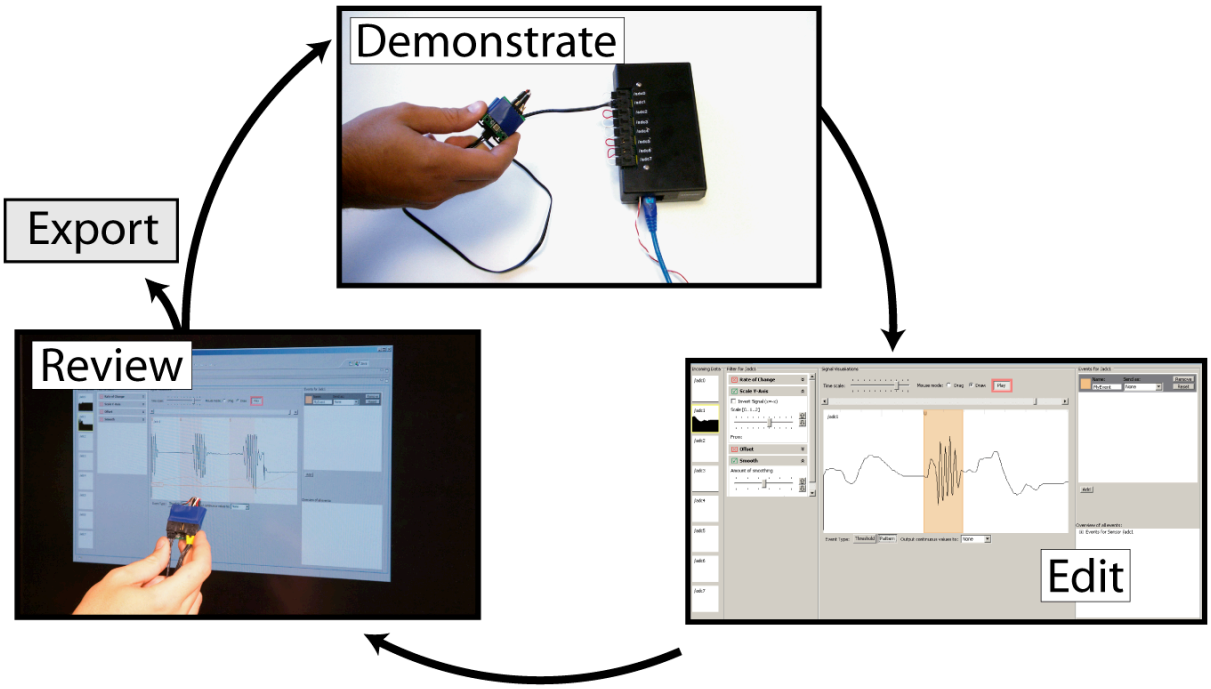


Figure 176 : Exemplar : Démonstration de l'interaction pour avoir des données exemples, éditer pour corriger ces données exemples, puis vérifier que ces corrections sont adéquates. Illustration extraite de [Hartmann et al., 2007].

Exemplar traite donc une facette de la conception qu'est la spécification des actions physiques sur un capteur. Il fournit donc un élément pour la conception qui peut être intégrée à d'autres outils de prototypage, y compris notre outil.

2.2.7. Juxtapose

Juxtapose [Hartmann et al., 2008b] est un outil qui facilite le prototypage simultané de différentes alternatives de conception. *Juxtapose* consiste en un éditeur de code (Figure 177) et un environnement d'exécution (Figure 178).

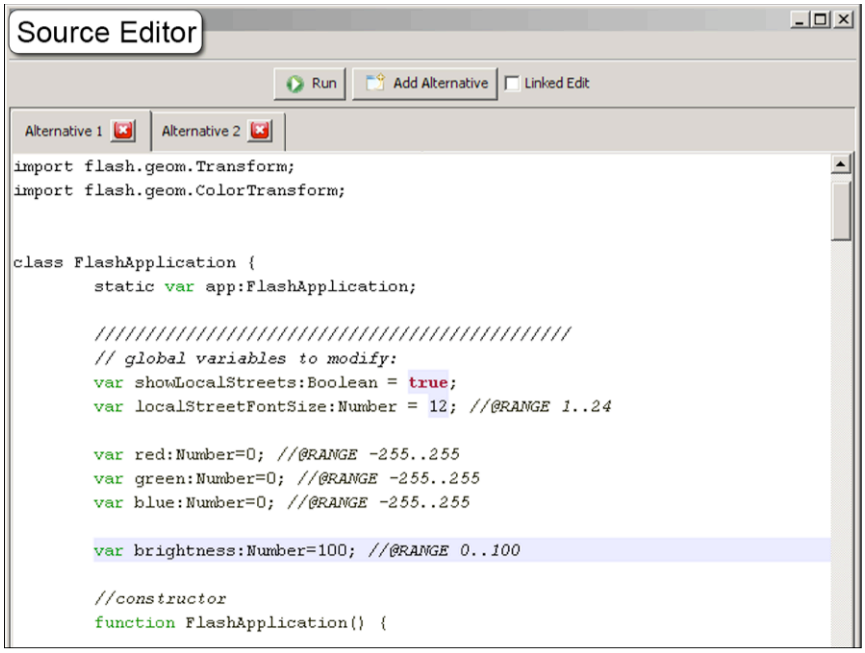


Figure 177 : Editeur de code de Juxtapose : l'exemple de deux alternatives pour la visualisation d'une carte. Une ligne de code est surlignée en bleu, signifiant que cette ligne diffère selon les prototypes. La solution alternative décrite consiste ici à fixer la luminosité à 100. Illustration extraite de [Hartmann et al., 2008b].

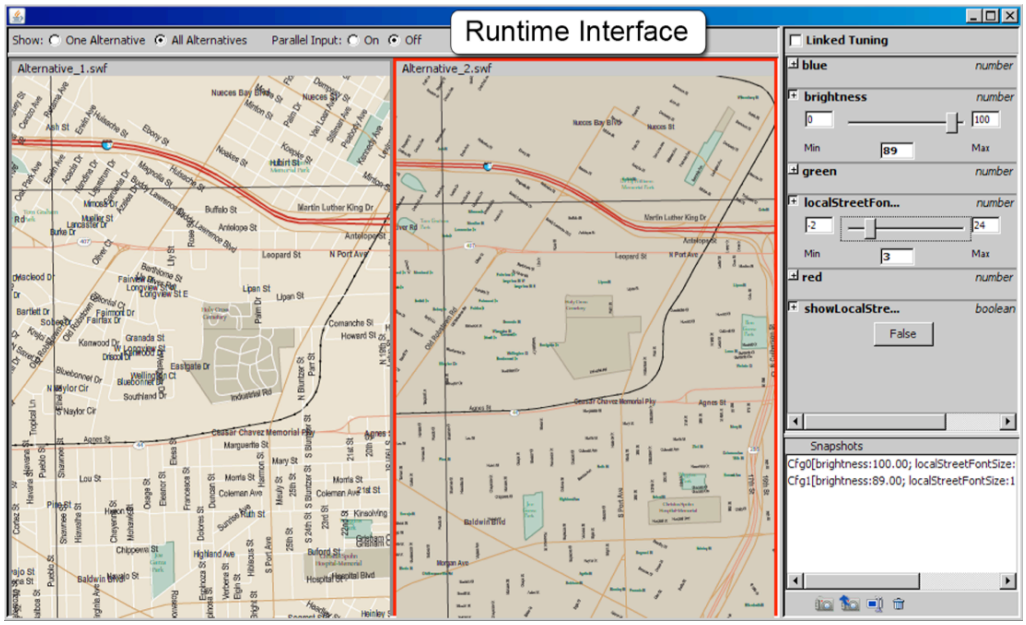


Figure 178 : Environnement d'exécution des alternatives dont une partie du code est présenté Figure 177. Par exemple, parmi les différences entre l'espace de travail de gauche et celui du milieu, nous trouvons la luminosité de la carte (Figure 177). À ce paramètre s'ajoutent d'autres paramètres, présentés et contrôlables dans l'espace de travail de droite de l'interface. Illustration extraite de [Hartmann et al., 2008b].

Pendant l'édition du code, le développeur peut spécifier s'il édite toutes les alternatives à la fois (modifications partagées par toutes les solutions), ou s'il édite uniquement l'alternative courante (onglet sélectionné Figure 177), en cochant ou ne cochant pas le champ « *Linked Edit* » (Figure 177 en haut à droite). Dans l'exemple de la Figure 177, le développeur a défini deux alternatives (deux onglets à la Figure 177), et trois paramètres (surlignés en bleu) qui varient entre ces deux alternatives : « showLocalStreets », « localStreetFontSize », et « brightness ».

Si le développeur appuie sur le bouton *Run* (Figure 177 en haut à gauche), alors l'interface d'exécution de la Figure 178 est générée.
Pendant l'exécution, le développeur peut visualiser les différentes alternatives de conception prototypées et ajuster les variables de ses prototypes simultanés pendant l'exécution (Figure 178). Les deux alternatives codées dans les deux onglets de l'éditeur de la Figure 177 sont présentées dans les deux tiers de gauche de l'écran de la Figure 178. Le tiers de droite de la Figure 178 présente l'interface pour ajuster les paramètres des prototypes.

Le développeur peut mémoriser les valeurs des paramètres à un instant donné de l'exécution, et recharger ces paramètres plus tard. Pour pouvoir ajuster les variables pendant l'exécution, *Juxtapose* propose un mécanisme basé sur la reconnaissance des commentaires qui suivent la déclaration des variables. Par exemple à la Figure 179, la déclaration de la variable « brightness » est suivie du commentaire « *//@RANGE 0..100* » (ligne 1), elle sera donc ajustable dans cet intervalle. Au contraire la variable « counter » est suivie du commentaire « *//@IGNORE* » et ne sera donc pas ajustable (ligne 2). De plus, la ligne 7 spécifie que la fonction de rappel (*callback*) sera appelée lorsque le concepteur ajustera la variable « brightness » grâce à l'interface graphique (Figure 178, partie droite de l'écran). Cette fonction est définie par le concepteur (ligne 3 à 6).

```
1 var brightness : Number = 100 ; //@RANGE 0..100
2 var counter ; //@IGNORE
3 var callback= function(varName, oldVal, newVal) {
4     redraw() ;
5     return newVal ;
6 }
7 this.watch('brightness', callback) ;
```

Figure 179 : Fragment de code montrant comment la variable « brightness » pourra être ajustée pendant l'exécution, et comment la variable « counter » ne le sera pas.

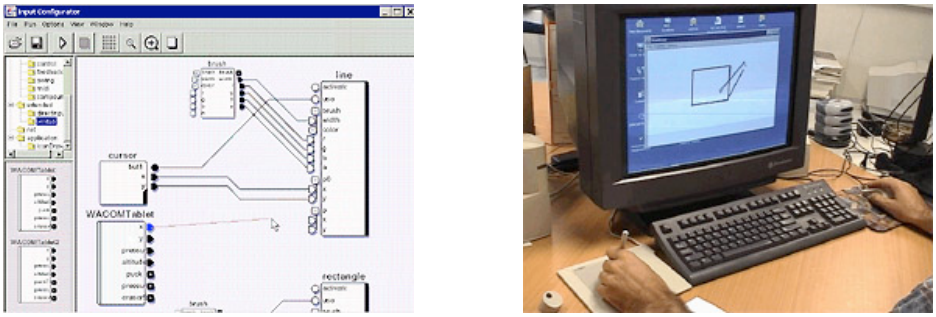
Juxtapose autorise des interfaces graphiques écrites en ActionScrip avec Flash (Adobe), sur téléphones mobiles avec Flash Lite, ainsi que des interfaces physiques ou tangibles avec Arduino (http://www.arduino.cc). La Figure 178 montre l'exemple de l'environnement d'exécution pour ordinateur de bureau et interface graphique. Ces trois implémentations différentes de Juxtapose partage le même éditeur (Figure 177), mais pas le même environnement d'exécution, qui comporte alors des différences et concessions par rapport à la version pour interface graphique sur ordinateur de bureau (Figure 178).

Comme pour Exemplar, Juxtapose traite une facette de la conception qu'est le prototypage simultané d'alternatives. Il fournit donc un élément pour la conception qui peut être intégrée à d'autres outils de prototypage, y compris notre outil.

2.2.8. ICON

ICON [Dragicevic, Fekete, 2004] vise l'adaptation des applications à différents dispositifs d'interaction en entrée du système. L'élément de base d'ICON est le composant (rectangles blancs de la Figure 180 (a)). Les composants peuvent être connectés :

- En entrée, pour recevoir de l'information (connexions blanches à gauche des composants dans la Figure 180 (a)), ou en sortie pour fournit de l'information (connexions noires à gauche des composants dans la Figure 180 (a)). Ces connexions sont typées, et peuvent être configurées. Un connecteur de sortie peut être connecté à plusieurs connexions d'entrée. Symétriquement plusieurs connecteurs de sortie peuvent être connectés à une seule connexion d'entrée.
- Explicitement ou implicitement. L'utilisateur peut connecter explicitement des éléments en les reliant. Au contraire, une connexion implicite est une connexion vers un élément qui n'est pas présent dans la configuration manipulée par le développeur, comme le curseur de la souris affiché sur l'écran.



(a) (b)

Figure 180 : Interface d'ICON (a) pour une interaction bi-manuelle (b).

ICON propose différents types de composants, disponibles dans la bibliothèque :

- Les plus à gauche dans l'espace de travail de la Figure 180 représentent les dispositifs (souris et tablette Wacom dans cet exemple). Quelques (peu de) dispositifs de sorties sont intégrés, par exemple la synthèse vocale.
- Les composants les plus à droite sont ceux qui représentent les entrées de l'application (« dispositifs d'application »). Ils permettent de contrôler des interfaces d'applications en Swing (http://java.sun.com/j2se/1.5.0/docs/guide/swing/). Ces applications peuvent être indépendantes d'ICON.
- Les composants intermédiaires sont des transformations et compositions entre les dispositifs et les applications, comme les opérateurs mathématiques (addition, soustraction, multiplication, division) et booléens (ET, OU), traitements du signal (filtres), adaptateurs de types (entier vers réel par exemple), ou opérateurs conditionnels (SI). Quelques composants pour un retour d'information graphique sont fournis, comme les curseurs et des cadres semi-transparents qui peuvent se superposer aux fenêtres.

Un ensemble de composants et leurs connexions peuvent être encapsulés dans un nouveau composant global.

Pour ajouter un composant à la librairie d'ICON, le développeur doit déclarer les connexions du composant, et implémenter une méthode appelée « update ».
Pour rendre une application sensible à ICON, il faut décrire la façon dont elle doit être contrôlée plutôt que de faire appel directement aux évènements clavier ou souris. Pour cela, il faut implémenter et déclarer un « dispositif d'application » (élément de la Figure 180 (a), sans sorties noires à sa droite). Comme pour ajouter un composant, le développeur doit implémenter une méthode « update » qui appelle les fonctions de l'application.

Le niveau d'abstraction d'ICON est trop élevé pour nous permettre de construire des objets mixtes. Comme dans d.tools, la structure proposée par ICON (description des transformations successives dans la trajectoire d'interaction) est différente de celle de notre modèle d'interaction mixte (description des objets mixtes prenant part à l'interaction). L'approche adoptée par ICON est plus macroscopique que l'outil que nous visons et un objet mixte pourrait être un composant dispositif dans ICON. Notre outil pourrait donc servir de fabrique à composants ICON.

2.2.9. ICARE et OpenInterface Framework

ICARE [Bouchet et al., 2004][Bouchet, 2006] est une plateforme à composants pour le développement des interfaces multimodales. Cette plateforme a été reprise, étendue et améliorée dans la plateforme OpenInterface [Serrano et al., 2008]. La plateforme OpenInterface se décompose en deux parties : le kernel OI (ou noyau), plateforme d'exécution, et l'OIDE, environnement de développement graphique pour l'interaction. La Figure 181 présente un exemple d'interaction avec une application prototypée avec la plateforme OpenInterface. L'utilisateur se sert d'un microphone pour commander le zoom, et de la souris pour spécifier le centre du zoom. Les dispositifs utilisés sont standards, et OI sert à prototyper l'interaction entre ces dispositifs et l'application. En bas de la Figure 181 est présenté l'éditeur graphique OIDE.

ICARE a été adapté pour le cas de la multimodalité en sortie [Mansoux, 2006]. ICARE ou la plateforme OI se concentrent que sur les modalités d'interaction et leurs compositions. Ces propositions se focalisent ainsi sur le prototypage de l'interaction avec l'application, et ne nous permettent pas de centrer le prototypage, comme la conception, autour des objets mixtes. Comme pour ICON, l'outil que nous visons est complémentaire et peut servir de fabrique à composants dispositifs d'OI.

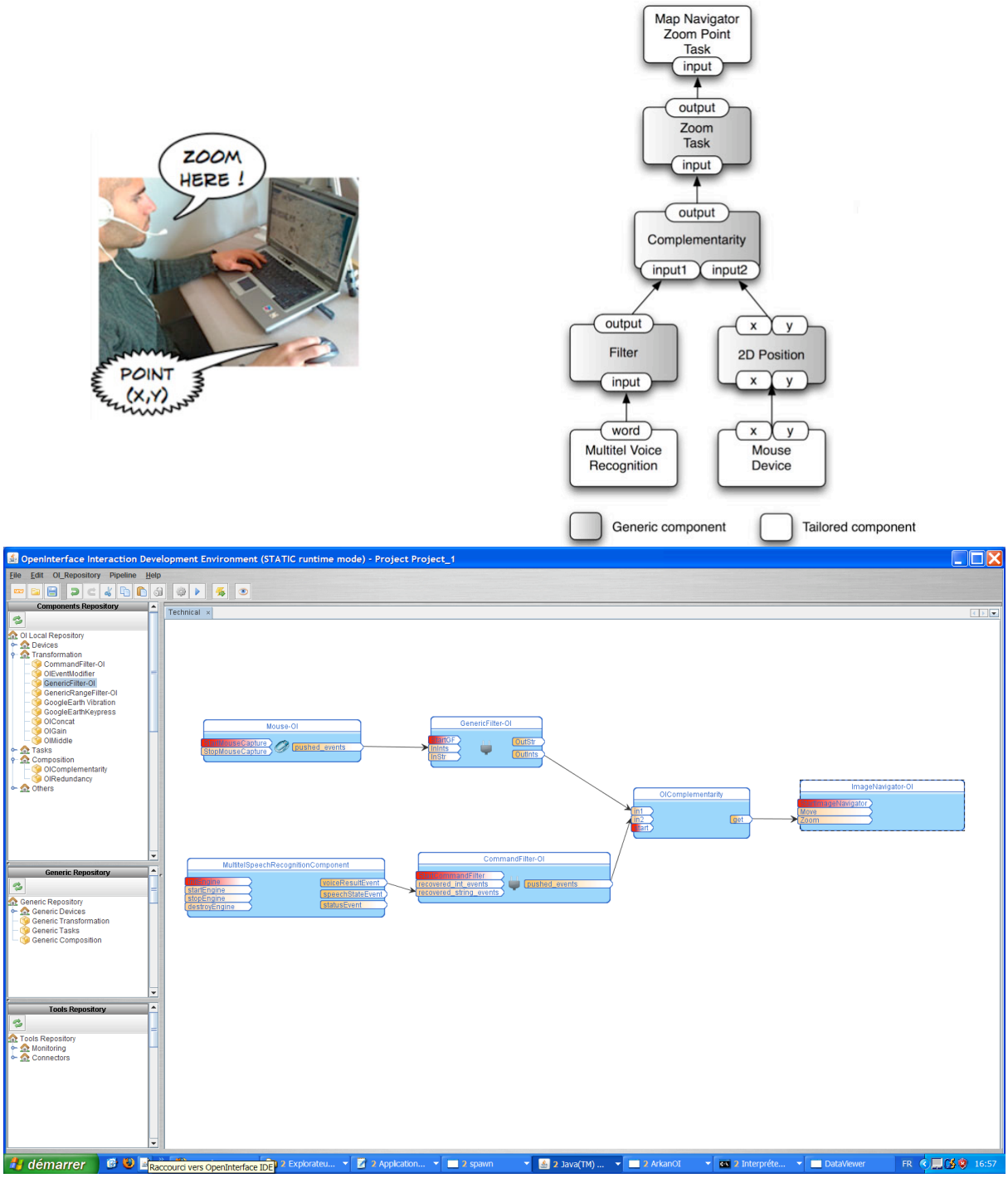


Figure 181 : Exemple d'interaction prototypée avec la plateforme OpenInterface. Illustration extraite de [Serrano et al., 2008].

2.2.10. ASUR-IL, WComp, GuideMe et Simba

ASUR [Dubois, 2001][Dubois, Gray, 2007] est une notation de conception que nous avons présenté au Chapitre 3 (section 1.1.8). Au-delà du modèle d'interaction, ASUR s'insère dans un processus d'ingénierie dirigée par les modèles. Pour aller de la conception des techniques d'interaction modélisées avec ASUR, jusqu'à l'implémentation du système, le processus s'appuie sur d'autres modèles plus proches de l'implémentation :

- ASUR-IL [Gauffre et al., 2007], sigle de « ASUR Implementation Layer », est un modèle pour la conception *logicielle*. Avec ce modèle, sont traitées les questions d'architecture logicielle déduite d'un diagramme ASUR, tout en restant encore indépendant de l'implémentation. La Figure 183 montre un modèle ASUR-IL pour le système modélisé en ASUR à la Figure 182. Le système considéré et modélisé à la Figure 182 permet à un utilisateur de construire un arbre de classification. Il manipule un objet physique (« *Species* ») représentant une espèce pour l'ajouter à l'arbre. Cet objet est équipé d'un marqueur comme avec l'ARToolKit. L'arbre de classification résultant est projeté sur une surface interactive. À la Figure 183, nous retrouvons les concepts du schéma ASUR de la Figure 182. Ainsi les concepts manipulés par ASUR sont explicites dans l'architecture logicielle, sous la forme d'éléments architecturaux : à gauche à la Figure 183 les modalités d'entrée, à droite les modalités de sortie, toutes deux reposant sur des adaptateurs d'ASUR, et au centre les éléments architecturaux MVC qui exploitent les composants systèmes d'ASUR. Ils se retrouvent dans les éléments d'architecture logicielle : en rouge les modalités d'entrée, en vert l'architecture MVC, en bleu les modalités de sorties.

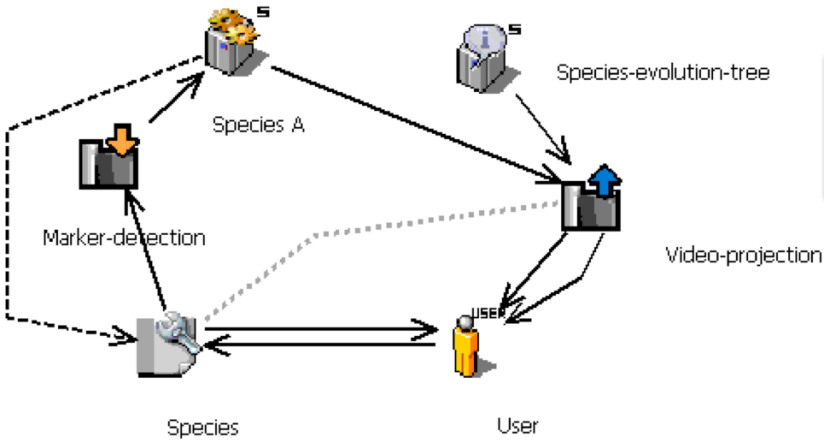


Figure 182 : Modèle ASUR d'un système de réalité mixte pour la construction d'un arbre de classification des espèces. Illustration extraite de [Gauffre et al., 2007].

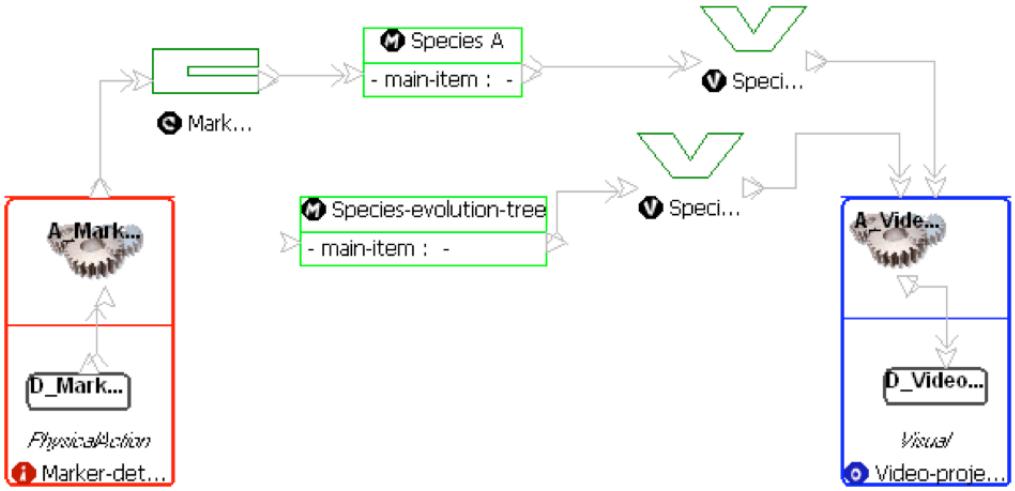


Figure 183 : Le modèle ASUR-IL correspondant au modèle ASUR de la Figure 182. Illustration extraite de [Gauffre et al., 2007].

- WComp [Cheung-Foo-Wo et al., 2007], une plateforme à composants pour le développement de dispositifs informatiques ubiquitaires et « wearable ». WComp permet l'adaptation au contexte pendant l'exécution (par exemple, si un dispositif est déconnecté). Au sein du processus d'ingénierie dirigée par le modèle, le passage d'un schéma ASUR-IL à un assemblage de composants WComp est une recherche en cours [Gauffre et al., 2007].

L'approche reposant sur ASUR consiste à définir un processus complet d'ingénierie dirigée par les modèles en partant d'un schéma ASUR. Pour cela des modèles plus proches de l'implémentation ont été considérés. Ainsi il est possible de passer d'un schéma ASUR à un schéma ASUR-IL puis bientôt d'ASUR-IL à l'implémentation avec WComp. Ce processus (ASUR → ASUR-IL → WComp) s'accompagne d'un outil logiciel de mise en ouvre, intitulé GuideMe (http://www.irit.fr/~Guillaume.Gauffre/ - projects). À cet ensemble d'outils s'ajoutent SIMBA [Abou Moussa et al., 2008] permettant la simulation logicielles des techniques d'interaction conçues.
Les travaux liés à ASUR répondent à des objectifs différents des nôtres. Tandis qu'ASUR est considéré comme une modélisation de départ dans un processus dirigé par les modèles pour le développement d'un système de réalité mixte, nos objectifs visent la conception d'un outil de prototypage rapide et non de développement.

Nous avons présenté les outils matériels et les outils logiciels. La disponibilité de ces outils est une condition nécessaire pour articuler ou comparer notre contribution aux travaux présentés ici. Nous présentons maintenant notre analyse de la disponibilité de ces outils.

2.3. Disponibilités

La question de la disponibilité se pose surtout pour les travaux issus de la recherche. Nous déclinons la disponibilité d'un outil selon plusieurs axes :

- la disponibilité au téléchargement pour un outil logiciel,
- la disponibilité de l'outil pour plusieurs plateformes (comme Windows, Linux, Mac OS)
- la maintenance de l'outil logiciel,
- la possibilité d'acheter l'outil dans le cas d'une plateforme matérielle,

Nous résumons dans le Tableau 25 les critères influençant la disponibilité des outils de prototypage que nous avons présentés. Pour chaque outil, nous notons si des ressources sont disponibles (colonne Diffusion), le dynamisme de la communauté de ses utilisateurs (colonne Dynamisme), ainsi que les plateformes matérielles et logicielles qu'il autorise (colonne plateforme matérielle et logicielle).

Outil	Diffusion	Dynamisme	Plateforme matérielle et logicielle	
			Système d'exploitation	Langage(s)
Capteurs et effecteurs divers	Matériel disponible à la vente.	→	Windows, MacOS, Linux	Protocole suffisamment répandu pour être largement couvert par les langages
Arduino	Matériel disponible à la vente. Logiciel téléchargeable gratuitement : http://arduino.cc/en/Main/Software Documentation disponible : http://arduino.cc/	↑	Windows, MacOS, Linux	Spécifique
Phidgets	Matériel disponible à la vente : http://www.phidgets.com/ Logiciel téléchargeable gratuitement : http://www.phidgets.com/ http://grouplab.cpsc.ucalgary.ca/cookbook/index.php/Toolkits/SharedPhidgets3 Documentation disponible : http://www.phidgets.com/ http://grouplab.cpsc.ucalgary.ca/cookbook/index.php/Toolkits/SharedPhidgets3	→	Windows, MacOS, Linux, Windows Mobile	.NET, Visual Basic, VBA (Microsoft Access et Excel), LabView, Java, Delphi, C, C++, Python
Switcharoo	Pas disponible Pas de documentation	↓	Windows, MacOS	
Calder	Pas disponible. Pas de documentation	↓		
BOXES	Pas disponible Pas de documentation	↓		
ARToolKit	Logiciel téléchargeable gratuitement : http://www.hitl.washington.edu/artoolkit/ Documentation disponible : http://www.hitl.washington.edu/artoolkit/	→	Windows, MacOS, Linux, iPhone	
Papier-Mâché	Logiciel téléchargeable gratuitement : http://hci.stanford.edu/research/papier-mache/ Documentation disponible : http://hci.stanford.edu/research/papier-mache/	↓ (Pas maintenu depuis juin 2004)	Windows	Java
iStuff	Logiciel téléchargeable gratuitement : http://hci.rwth-aachen.de/istuff/	→	Windows, MacOS, Linux	Java
iStuff mobile	Pas disponible Pas de documentation	→	MacOS, Symbian Series 60	Quartz Composer

Context Toolkit	Logiciel non disponible directement sur le web : il faut contacter l'auteur pour obtenir la boîte à outils.	↓		
d.tools	Logiciel téléchargeable gratuitement : http://sourceforge.net/projects/d-tools/ Documentation disponible : http://sourceforge.net/projects/d-tools/	→	Windows (Limité aux PC windows, à cause de la vidéo et de l'USB)	
Exemplar	Logiciel téléchargeable gratuitement : http://hci.stanford.edu/research/exemplar/ Documentation disponible : http://hci.stanford.edu/research/exemplar/	→		
ICON	Logiciel téléchargeable gratuitement : http://inputconf.sourceforge.net/	→	Windows, MacOS, Linux (Java, mais dispositifs particuliers pour Windows et Linux)	
ICARE + OI	Logiciel téléchargeable gratuitement http://oi-project.org/	→	Windows	Java, C++, Matlab, Python, .NET
ASUR-IL, WComp et GuideMe	WComp téléchargeable gratuitement : http://rainbow.essi.fr/wikiwcomp/doku.php Documentation pour WComp disponible : http://rainbow.essi.fr/wikiwcomp/doku.php	→		

Tableau 25: Disponibilité des outils étudiés (↓ : Faible dynamisme des ressources, → : Dynamisme des ressources, ↑ : Fort dynamisme des ressources).

3. Analyse des outils existants au regard des objectifs

Nous rappelons que nos objectifs se situent à deux niveaux :

1. En considérant le prototypage de façon isolée, notre objectif est de permettre le prototypage rapide de l'interaction, avec une grande résolution (niveau de fidélité du prototype) et envergure (nombre de dimensions de conception explorées par le prototype).
2. En considérant le prototypage comme l'aspect matérialisation de la conception, notre objectif est d'opérationnaliser notre modèle d'interaction mixte présenté au Chapitre 4 pour mieux articuler exploration conceptuelle et pratique.

Nous analysons maintenant les outils présentés au regard de ces deux niveaux.

En considérant le prototypage de façon isolée, nous visons tout d'abord à prototyper très tôt à la fois l'interaction accompagnée de l'apparence, pour augmenter l'envergure des prototypes. C'est un objectif partagé avec d'autres outils, comme Switcharoo, Calder ou BOXES qui le revendiquent

explicitement. Pourtant, BOXES propose un nombre limité d'interactions possibles (équivalentes au clavier et à la souris), et, comme Switcharoo et Calder, il propose donc une solution locale pour augmenter la résolution des premiers prototypes.

Nous visons aussi l'augmentation de la résolution des premiers prototypes en baissant le coût du prototypage de l'interaction. Pour baisser le niveau de difficulté de prise en main (« seuil[42] » de [Myers et al., 2000]), certains outils comme d.tools proposent une interface graphique. Nous soulignons que la présence d'un interface graphique n'est pas en relation directe avec l'expertise des utilisateurs cibles. En effet, un outil comme Arduino propose une interface textuelle (code) à ses utilisateurs non-informaticiens.
Pour baisser le coût de prototypage de l'interaction, certains outils offrent la possibilité à des novices de prototyper avec une technologie difficile d'accès, comme l'électronique (les Phidgets par exemple) ou la vision par ordinateur (l'ARToolkit par exemple). D'autres outils facilitent l'intégration de dispositifs non-standards dans les applications existantes, comme ICON ou OpenInterface. Pourtant, tous ces outils qui diminuent la difficulté de prise en main sont dédiés à un type particulier de ressources matérielles (dispositifs pré-existants souris, tablettes, graphiques, etc. pour ICON, ou à construire avec les Phidgets) ou ne permettent de prototyper que l'interaction en entrée (context toolkit, OpenInterface, ICON). Par conséquent, leurs possibilités sont limitées (« plafond[43] » de [Myers et al., 2000]). Ils ne peuvent donc augmenter la résolution des prototypes que localement, pour le problème particulier qu'ils résolvent.

Nous visons aussi à permettre une évolution rapide en facilitant les modifications dans le cadre de micro-itérations. Certains outils répondent à cet objectif, comme d.tools ou Exemplar. Pour cela, plusieurs outils proposent une interface graphique pour corriger les erreurs et la simulation en magicien d'Oz (Phidgets, Papier-Mâché, d.tools, IStuff mobile). Ces outils répondent donc à un de nos objectifs.

Enfin, nous visons l'adaptation aux innovations technologiques existantes et à venir. Par exemple, l'ARtoolkit a été étendu avec succès et d.tools a intégré Arduino et les Phidgets. Au contraire, d'autres outils n'ont pas été étendus et ont été abandonnés, comme Papier-Mâché (Tableau 25).

À plus grande échelle, en repositionnant le prototypage dans la conception comme nous l'avons fait dans l'introduction, nous visons la matérialisation des alternatives issues de l'exploration de l'espace de conception avec le modèle d'interaction mixte. L'outil doit permettre de rebondir entre activités conceptuelles et activités pratiques du prototypage en définissant une transition fluide et directe entre les solutions modélisées et leur matérialisation. Avoir un outil de prototypage qui opérationnalise notre modèle est une condition nécessaire à l'adoption de notre modèle d'interaction [Beaudoin-Lafon, 2004].
Parmi les outils de prototypage présentés, peu se positionnent dans le contexte de l'activité de conception. Seul d.tools explicite qu'il prend en compte plusieurs étapes de la conception (prototypage, évaluation, analyse). Pourtant, il n'est pas explicitement basé sur un modèle d'interaction et ne prend pas en compte la phase conceptuelle. Il repose donc sur l'expertise de ses utilisateurs pour explorer efficacement l'espace des possibilités et ne répond donc pas à nos objectifs.
Pour répondre à notre objectif de prototype comme matérialisation de la conception, il faut notamment un outil qui offre une forte modularité au niveau d'abstraction correspondant à l'objet mixte, intermédiaire entre dispositifs physiques (ressources matérielles) et applications. En effet, si ces objets sont le centre de la conception avec le modèle d'interaction mixte (Chapitres 3 et 4), ils ne peuvent pas être dissous dans le prototype. Or nous avons souligné lors de cette revue des outils de prototypage qu'aucun ne propose un prototypage centré sur l'objet mixte.

[42] threshold
[43] ceiling

De ces différences dans l'approche du prototypage rapide, nous retenons également des outils et mécanismes qui peuvent être complémentaires et orthogonaux à notre outil : permettre la reconfiguration pendant l'exécution (IStuff) ou encore proposer de lier le prototypage à l'évaluation (d.tools). Enfin les travaux basés sur ASUR sont différents car ils traitent le développement d'un système de réalité mixte et non du prototypage. Nous retenons de ces travaux que notre modèle d'interaction mixte pourrait être un des modèles mis en jeu dans un processus d'ingénierie dirigé par les modèles dédiés aux systèmes de réalité mixte.

4. Synthèse

En synthèse, nous avons organisé notre revue des outils de prototypage selon deux catégories.
D'un côté, nous trouvons les outils « bas niveau » qui permettent de résoudre une difficulté technologique attachée au lien entre le monde physique et numérique. Par exemple les Phidgets rendent disponibles des capteurs prêts à l'emploi avec une API, pour ceux qui ne sont pas experts en électronique. Ces boîtes à outils (Phidgets, ARToolKit, BOXES, IStuff, Exemplar, Papier-Mâché, Context Toolkit, etc.) sont utiles pour prototyper des objets mixtes, néanmoins ils n'explicitent pas les différents niveaux d'abstraction identifiés dans notre modèle d'interaction pour concevoir un objet mixte. Avec ces outils, il existe alors un fossé entre la conception d'un objet mixte modélisé selon notre modèle et sa matérialisation grâce à l'outil.
Nous trouvons d'un autre côté des outils de prototypage « haut niveau » qui permettent de prototyper l'interaction dans son ensemble, comme d.tools, ICON, ICARE et OI. Ces outils ne proposent pas l'objet mixte comme composant élémentaire de la construction des interfaces. Néanmoins, certains d'entre eux, comme ICON et OI, pourraient accueillir des objets mixtes prototypés par d'autres moyens.

Notre étude des outils existants motive la définition et le développement d'un outil reposant explicitement sur notre modèle d'interaction. Notre outil devra intégrer les outils que nous avons classés comme « bas niveau » et être une fabrique d'éléments de base pour les outils classés comme « haut niveau ». Le chapitre suivant présente cet outil de prototypage.

Chapitre 7 : La solution proposée, la boîte à outils OP

Au Chapitre 6, nous avons exposé notre objectif en termes d'outils de prototypage. Celui-ci est double :

1. Prototypage rapide avec une haute résolution et envergure en considérant l'interaction et l'apparence,
2. Opérationnalisation du modèle d'interaction mixte afin de combiner efficacement activités de modélisation et de prototypage lors de la conception. Comme pour le modèle d'interaction mixte, avant même le prototypage d'interaction nouvelles, notre objectif est la capacité de prototypage de solutions issues de l'exploration systématique de l'espace de conception.

Dans ce chapitre, nous présentons la boîte à outils appelée OP (*Object Prototyping*[44]) qui, tout en s'appuyant sur les outils existants étudiés au chapitre précédent, répond à notre objectif. Dans cette boîte à outils, les éléments à la disposition du développeur correspondent aux éléments du modèle d'interaction mixte. En effet, les évaluations du modèle d'interaction mixte (Chapitre 5) montrent le besoin de changements au niveau extrinsèque (changer d'objet mixte dans l'interaction avec l'application) et au niveau intrinsèque (changer les éléments constituant un objet mixte). La boîte à outils OP propose donc des éléments logiciels pour construire le prototype d'un objet mixte (niveau intrinsèque).

La structure de ce chapitre est la suivante : nous décrivons d'abord la boîte à outils dans son ensemble. Nous présentons ensuite les éléments OP qui sont disponibles aujourd'hui. Nous montrons ensuite sur l'exemple d'un objet sensible à la lumière comment utiliser OP pour prototyper un objet mixte. Enfin, nous présentons comment insérer dans une application un objet mixte prototypé avec la boîte à outils OP.

5. Description de la boîte à outils OP

La boîte à outil OP repose explicitement sur le modèle d'interaction mixte. Ainsi les éléments OP qui composent la boîte à outils correspondent aux concepts et niveaux d'abstraction manipulés dans notre modèle. Avant de les décrire, nous présentons comment les éléments sont réalisés, décrits par des propriétés et connectés entre eux. Nous définissons ensuite les nomenclatures et conventions de la boîte à outils avant de décrire la structure de la boîte à outils qui repose sur notre modèle. Dans la description d'OP, nous notons les éléments de la boîte à outils des **composants**, au sens d'*éléments constitutifs de la boîte à outils*. Dans notre contexte, le terme « objet » prête à confusion avec l'objet mixte à prototyper.
Pour cela, nous présentons d'abord l'origine des composants OP. Ensuite, nous expliquons ce que sont les propriétés des composants OP, comment connecter entre eux les composants d'un objet mixte, les nomenclatures et conventions de la boîte à outils, et sa structure. Nous présentons enfin son outil annexe : l'interface graphique pour corriger et simuler un objet mixte.

5.1. Éléments Constitutifs : Composants OP

Les composants de la boîte à outils OP sont basés sur Qt (http://www.qtsoftware.com). La boîte à outils Qt est notamment utilisée avec C++ pour le développement des interfaces graphiques. Outre son utilisation répandue, les caractéristiques de Qt qui sont pertinentes pour OP sont :

- Des **propriétés** pour les *QObject* (http://doc.trolltech.com/4.5/properties.html) ;
- Un mécanisme de communication entre les *QObject*[45] : les **signaux** et les **slots**[46] (http://doc.trolltech.com/4.5/signalsandslots.html) ;

Ces deux caractéristiques sont rendues possibles dans Qt grâce au Compilateur Meta-Object (**moc**). L'outil moc analyse les fichiers d'entête (.h). Si moc trouve une ou plusieurs déclarations de classes qui contiennent la macro Q_OBJECT, alors il produit un fichier source C++ (.cpp) contenant un

[44] Prototypage d'Objet
[45] Objet Qt
[46] Ouvertures

code meta-object pour ces classes. Ce code meta-object est utilisé pour le mécanisme de communication des signaux et des slots, et le système des propriétés dynamiques. Le fichier source C++ généré par moc doit être compilé et lié avec l'implémentation de la classe initiale.
Ainsi tous les composants OP héritent de QObject pour avoir des signaux, des slots et des propriétés. Nous présentons maintenant quelles sont les propriétés des composants OP.

5.1.1. Propriétés d'un composant

Les propriétés sont des membres[47] de la classe d'un composant qui sont importantes pour l'utilisateur-développeur : ce sont les caractéristiques du composant, grâce auxquelles un composant générique peut être adapté par l'utilisateur-développeur. Par exemple, la valeur de seuil du composant *ThresholdInputLanguage* est une propriété. Ainsi l'utilisateur-développeur utilise ce composant pour appliquer un seuil de la valeur de son choix en modifiant cette propriété.
Les membres sont déclarés comme propriétés grâce à la macro Q_PROPERTY. On peut ensuite accéder et/ou modifier leurs valeurs dynamiquement via le système des propriétés Qt (http://doc.trolltech.com/4.2/properties.html).
Ce système des propriétés de Qt est utilisé pour connaître les caractéristiques du composant pour les outils annexes. Nous envisageons également d'utiliser ce mécanisme pour permettre l'ajustement dynamique des caractéristiques des composants pendant l'exécution du prototype afin de rendre les itérations plus faciles et rapides pour une mise au point d'un objet mixte plus efficace.

5.1.2. Connexions entre les composants d'un objet mixte

Les composants OP peuvent être connectés entre eux grâce au mécanisme des signaux et des slots de Qt : un signal est émis lorsqu'un évènement particulier a lieu. Un slot est une fonction qui est appelée en réponse à un signal, si elle a été préalablement connectée à ce signal. La Figure 184 présente un exemple de connexions entre quatre objets. Par exemple dans ce cas, lorsque le signal 1 de l'objet 1 est émis, les slots 1 et 2 de l'objet 2 sont exécutés.

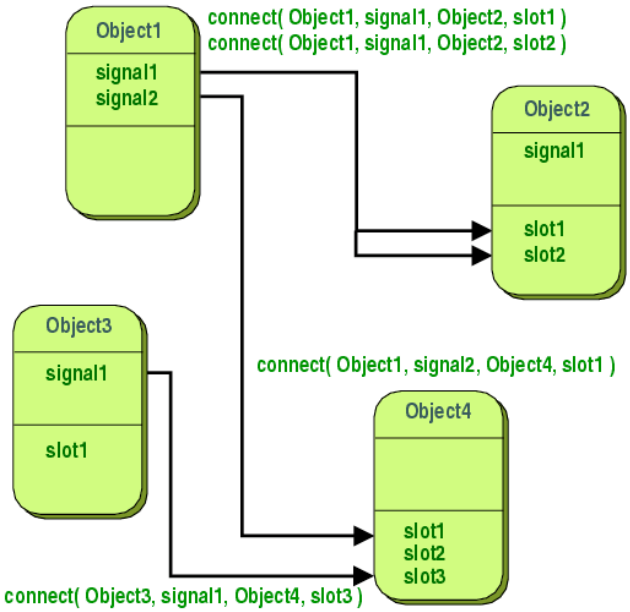


Figure 184 : Mécanisme des signaux et des slots de Qt. Illustration extraite de http://doc.trolltech.com/4.4/signalsandslots.html.

[47] Membres, attributs, champs, ou propriétés : *caractéristiques* d'un objet au sens de la programmation orientée objet, opposées à son *comportement* défini par les méthodes ou fonctions.

Le mécanisme des signaux et des slots est typé : la signature d'un signal doit correspondre à la signature d'un slot pour pouvoir y être connecté. Pourtant, la signature d'un slot peut être plus courte que celle du signal : celui-ci ignore alors les arguments en trop.

Il nous a été possible de définir nos propres signaux et slots pour les composants OP, car ils héritent de QObject. Dans OP, la sortie d'un composant est un signal, et l'entrée d'un composant est un slot. Connecter une sortie à une entrée se fait grâce à une ligne de code comme la suivante :

```
connect(OPComponent1, SIGNAL(updated), OPComponent2, SLOT(update))
```

Les composants OP ont, de façon prédéfinie, des signaux nommés *updated*, émis à chaque fois que la valeur de sortie du composant est modifiée, et des slots nommés *update*. Ces slots permettent de connecter la sortie d'un autre composant avec l'entrée du composant OP.

5.1.3. Nomenclature

Nous introduisons la nomenclature de la boîte à outils OP, pour en faciliter sa compréhension et son extension.
Les composants sont nommés selon la forme suivante : <Particularité>(<Direction>)<Niveau d'abstraction>. Le niveau d'abstraction issu du modèle d'interaction mixte peut être *Composition*, *Device*, ou *Language*. La direction n'est pas spécifiée si le composant peut servir pour l'entrée et la sortie. En revanche, s'il n'est utilisable que pour l'entrée ou que pour la sortie, alors la direction est spécifiée : *Input* ou *Output*. Enfin, la particularité indique qu'elle est sa spécificité par rapport aux autres composants de son niveau d'abstraction. Ainsi nous trouvons *MIDIDevice*, *ARVideoInputDevice* ou encore *DelayLanguage*. Le composant *DigitalProperty* n'a quant à lui pas besoin d'être étendu car il peut être utilisé pour différents types de propriétés. Il porte donc un nom général.
Les signaux d'un composant sont nommés *updated* et les slots *update*. La signature de ces signaux et slots doit être *(QVariant, QTime)* (dans la mesure du possible), afin de propager la valeur avec un type générique et une estampille temporelle.

Nous avons décrit comment sont réalisés et connectés les éléments constitutifs de la boîte à outils OP, notés composants OP. Nous présentons maintenant la structure globale de l'outil OP en termes de composants OP.

5.2. Structure de la boîte à outils

La Figure 185 présente la structure de la boîte à outils OP sous forme de diagramme d'héritage. La classe mère de tous les composants OP est *MixedObjectComponent* (Figure 185). Deux classes héritent directement de celle-ci : *LinkingComponent* et *DigitalProperties* (Figure 185).
La classe *DigitalProperty* qui hérite de *DigitalProperties* est un élément qui peut être utilisé pour construire une propriété numérique d'un objet mixte. De la classe *LinkingComponent* héritent trois classes filles : *Composition*, *Device*, et *Language* (Figure 185). Toutes les classes qui héritent de ces trois classes sont respectivement des composants de composition, de dispositifs et de langages de liaison qui peuvent êtres utilisés pour construire un objet mixte.

Ajouter un composant à la boîte à outils impose d'écrire le code d'une classe héritant de *Composition*, *Device*, ou *Language*. Nous avons pensé le composant *DigitalProperty* pour qu'il puisse servir pour un grand nombre de propriété numérique (par exemple des images, une valeur entière, une valeur booléenne, etc.) : le type des propriétés numériques est basé sur le type QVariant de Qt (http://doc.trolltech.com/4.2/qvariant.html - Type-enum). Il n'est donc a priori pas nécessaire de l'étendre.

La classe mère de tous les composants est *QObject*. OP est donc une extension de Qt. La propriété *objectName* de QObject (http://doc.trolltech.com/4.2/qobject.html - objectName-prop) permet de donner un nom au *QObject*, donc par héritage à tous les composants de OP.

Le composant *MixedObjectComponent* (Figure 185) a comme propriété le type du composant (*componentType*) : il peut prendre les valeurs :

- MixedObjectComponent::LINKINGCOMPONENT,
- MixedObjectComponent::DIGITALPROPERTIES.

Cette propriété est indiquée par l'utilisateur d'OP : *MixedObjectComponent (char *pName, ComponentType pComponentType)*.

Le composant *DigitalProperties* (Figure 185) a comme propriétés :

- *digitalPropertiesType* qui peut prendre entre autres la valeur *DigitalProperties::GENERIC* pour un composant adaptable à plusieurs types de propriétés numériques, comme des valeurs entières, booléennes, des images, etc. (http://doc.trolltech.com/4.2/qvariant.html - Type-enum). Les autres valeurs possibles correspondent à des composants sur mesure, ce que nous avons évité depuis les premières évaluations de la boîte à outils (Chapitre 8).
- *isBounceBack* qui détermine si la propriété nécessite un minuteur (*timer*) associé. Ceci permet de fixer la caractéristique « à rebond » des propriétés numériques (Chapitre 4 section 2.1.3.2 page 94).

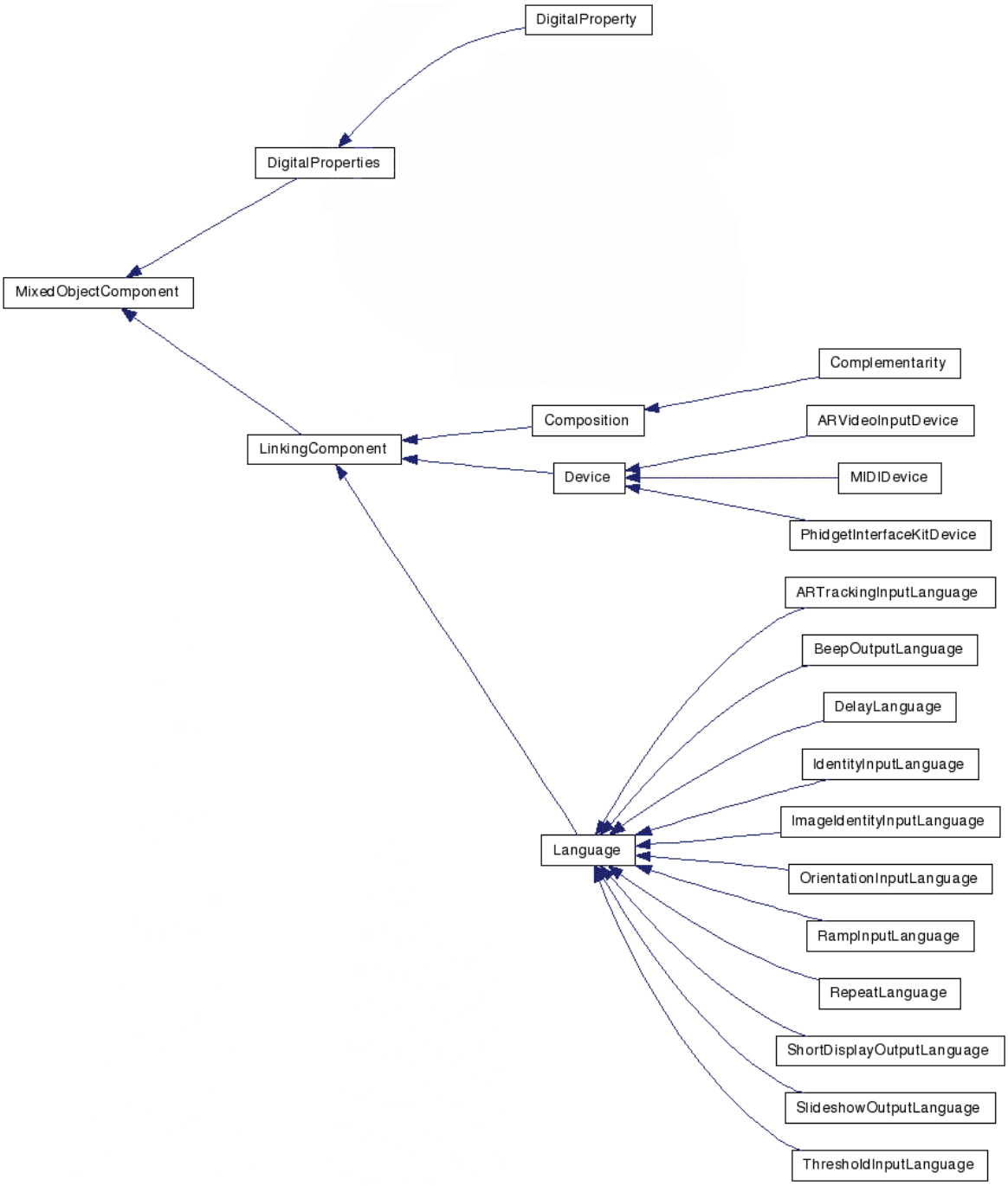


Figure 185 : Structure de la boîte à outils OP : diagramme d'héritage des composants OP.

Le composant *LinkingComponent* (Figure 185) a comme propriétés :

- *_linkingComponentDirection* qui peut prendre les valeurs :
  - LinkingComponent::IN,
  - LinkingComponent::OUT
- *_linkingComponentType* qui peut prendre les valeurs :
  - LinkingComponent::DEVICE,
  - LinkingComponent::COMPOSITION,
  - LinkingComponent::LANGUAGE

Il s'agit des différents types de composants héritant de *LinkingComponent* (Figure 185) et basés sur le modèle d'interaction mixte.

Le composant *Device* (Figure 185) a comme propriété *deviceType*, qui peut prendre les valeurs :

- Device::MIDI,
- Device::ARVIDEO,
- Device::PHIDGET

Il s'agit des différents types de composants héritant de *Device* (Figure 185), à la disposition de l'utilisateur d'OP.

Le composant *Composition* (Figure 185) a comme propriétés :

- *CompositionType* qui ne peut prendre aujourd'hui que la valeur *Composition::COMPLEMENTARITY*. Ces possibilités seront étendues dans le futur avec d'autres types de composition.
- *AbstractionLevel* qui peut prendre les valeurs *Composition::DEVICE* ou *Composition::LANGUAGE*. Cette propriété détermine à quelle niveau d'abstraction la composition est effectuée.
- *DeltaT* représente la fenêtre temporelle dans le cas de la composition temporelle de modalités (la seule composition disponible aujourd'hui).

Le composant *Language* (Figure 185) a comme propriété *LanguageType*, qui peut prendre les valeurs :

- Language::IN_IDENTITY
- Language::IN_THRESHOLD
- Language::REPEAT
- Language::IN_ORIENTATION
- Language::IN_ARTRACKING
- Language::IN_IMAGEIDENTITY
- Language::OUT_BEEP
- Language::OUT_SLIDESHOW
- Language::OUT_SHORTDISPLAY
- Language::IN_RAMP
- Language::DELAY

Il s'agit des différents types de composants héritant de *Language* (Figure 185), à la disposition de l'utilisateur d'OP.

La structure de la boîte à outils et les propriétés regroupées dans les classes mères permettent, entre autres, de rendre possibles une gestion simple de l'outil graphique de mise au point que nous présentons maintenant.

5.3. Outil graphique de mise au point d'un objet mixte

La boîte à outils OP propose un outil annexe au prototypage d'objets mixtes. Il s'agit d'une interface graphique optionnelle pour mettre au point un prototype et/ou simuler des parties.

Prenons l'exemple d'un objet mixte contenant une partie molle équipée d'un capteur de flexion et une partie dure équipée d'un écran (Figure 186). Lorsque l'utilisateur plie/relâche la partie molle au-delà d'un certain seuil, l'état interne de l'objet est modifié et son nouvel état est affiché sur l'écran : 1 s'il est au-dessus du seuil, 0 sinon. La description de cet objet selon le modèle d'interaction mixte est présentée à la Figure 186.

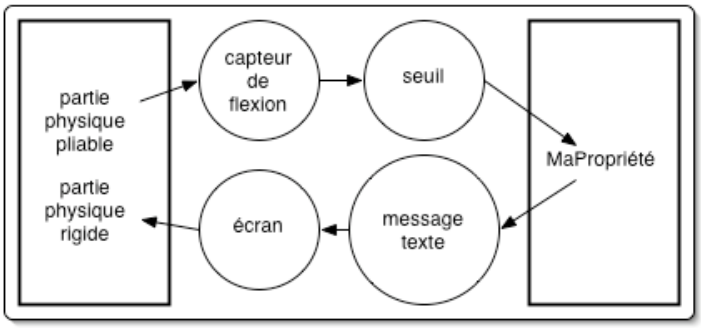


Figure 186 : Modélisation d'un objet mixte pliable.

Cet objet est prototypé avec les composants suivants de la Figure 185 :

- Un MIDIDevice appelé BendSensor,
- Un ThresholdInputLangage appelé Threshold,
- Une DigitalProperty booléenne MyProperty,
- Une ShortDisplayOutputLanguage appelé TextMessage.

Le dispositif est standard (l'écran), et nous n'avons donc pas besoin de dispositif OP de sortie.

La Figure 187 présente le code nécessaire pour construire l'interface de mise au point de cet objet mixte. Nous construisons un *MainWidget* (Figure 187, ligne 11 à 20, en gras) qui prend comme paramètres une référence au widget parent (aucun (0) dans ce cas) et des vecteurs contenant les composants de l'objet (Figure 187, ligne 1 à 10). À la ligne 22 du code, l'interface graphique de cet objet est affichée.

```
1  QVector<Device*> pInputDevice;
2  pInputDevice.push_back(&BendSensor);
3  QVector<Composition*> pCompositionEmpty;
4  QVector<Language*> pInputLanguage;
5  pInputLanguage.push_back(&Threshold);
6  QVector<DigitalProperty*> pDigitalProperties;
7  pDigitalProperties.push_back(&MyProperty);
8  QVector<Language*> pOutputLanguage;
9  pOutputLanguage.push_back(&TextMessage);
10 QVector<Device*> pOutputDeviceEmpty;
11 MainWidget ObjectWidget(0,
12                          pInputDevice,
13                          pCompositionEmpty,
14                          pInputLanguage,
15                          pCompositionEmpty,
16                          pDigitalProperties,
17                          pCompositionEmpty,
18                          pOutputLanguage,
19                          pCompositionEmpty,
20                          pOutputDeviceEmpty);
21 ObjectWidget.setWindowTitle("Object");
22 ObjectWidget.show();
```

Figure 187 : Construction de l'interface graphique pour la mise au point de l'objet pliable modélisé à la Figure 186.

Il est inutile de connecter le MainWidget aux composants de l'objet mixte (*BendSensor*, *Threshold*, *MyProperty*, *TextMessage)*, ni de préciser le type des composants *BendSensor*, *Threshold*, *MyProperty*, *TextMessage* pour que le MainWidget y soit connecté et propose la forme (glissière, image, ou boîte à cocher par exemple Figure 188). Le MainWidget est créé avec comme paramètre ces composants *BendSensor*, *Threshold*, *MyProperty*, *TextMessage* pour prendre la forme adéquate. Il y est aussi automatiquement connecté. En effet, le type des composants *BendSensor*, *Threshold*, *MyProperty*, *TextMessage* et leurs caractéristiques sont contenus dans les composants eux-mêmes, grâce aux propriétés de Qt décrites précédemment.
La Figure 188 présente l'interface graphique qui s'affiche lorsque la ligne 22 du code de la Figure 187 est exécutée. Chaque espace de l'interface correspond à un composant de l'objet.

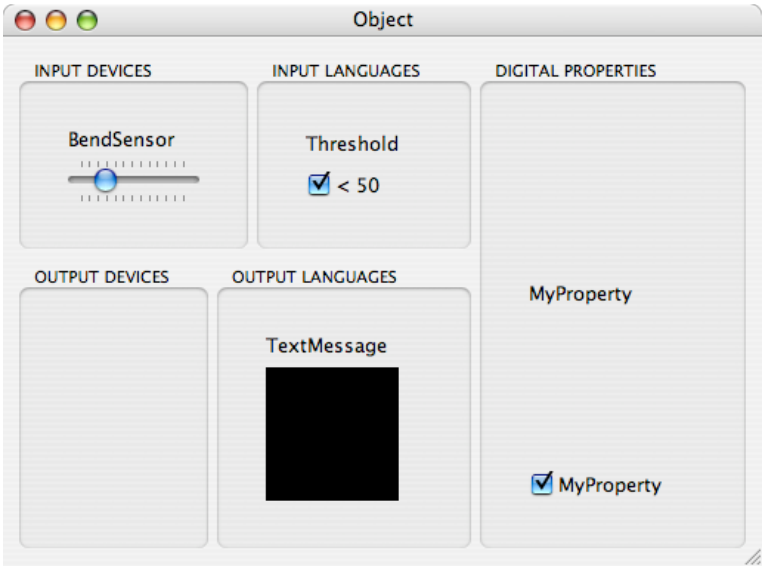


Figure 188 : Interface graphique de mise au point d'un objet : exemple d'un objet appelé *Object* qui capte la flexion parmi ses propriétés physiques, et met sa propriété numérique *MyProperty* à vrai ou faux selon que le capteur de flexion envoie une valeur respectivement au dessous ou au dessus de 50. Comme retour d'information, cet objet fournit un message affichant la valeur de *MyProperty* (0 ou 1) sur un écran.

Pour obtenir cette interface, nous avons développé un widget sur mesure adapté à chaque type de composants. Lorsque l'utilisateur d'OP construit un *MainWidget*, l'interface est construite selon les composants intrinsèques de l'objet mixte.
Cette interface peut être utilisée pour faire des corrections de mise au point (*debug*) et de la simulation de type Magicien d'Oz : elle permet d'interagir avec chaque composant du prototype. Dans notre exemple d'objet pliable, lorsque l'utilisateur plie l'objet et donc le capteur de flexion, l'expérimentateur voit la glissière sous *BendSensor* (Figure 188) bouger en accord avec ce mouvement. De même, si l'expérimentateur manipule la glissière sous *BendSensor* (Figure 188) avec la souris, alors l'objet mixte se comporte exactement de la même façon que si c'était l'objet que l'utilisateur avait plié. Ceci peut être utile si par exemple les concepteurs veulent expérimenter une idée sans avoir le matériel à disposition, ou encore si le capteur a été débranché par accident pendant une évaluation. Cette interaction que nous venons de décrire avec la glissière sous *BendSensor* (Figure 188) est possible avec tous les niveaux d'abstraction, via les widgets d'interaction graphiques de l'interface.

Pour instrumenter l'évaluation et enrichir les outils liés à OP, un mécanisme de génération de traces logicielles est également en cours de réalisation.

Nous venons de décrire la structure et la réalisation logicielle de la boîte à outils OP. Nous présentons maintenant les composants disponibles pour que l'utilisateur d'OP puisse prototyper des objets mixtes.

6. Composants OP disponibles

Afin de permettre une transition fluide entre l'utilisation du modèle d'interaction mixte et le prototypage, la boîte à outils OP offre une bibliothèque extensible de différents types de composants basés sur le modèle d'interaction mixte présenté au Chapitre 4. La boîte à outils offre des composants pour les dispositifs de liaison en entrée et en sortie, pour les langages de liaison en entrée et en sortie, pour la composition des modalités de liaison au niveau des dispositifs ou des langages, ainsi que des composants pour les propriétés numériques (Figure 185). Ce faisant, la boîte à outils vérifie les propriétés logicielles suivantes :

1. La modularité au niveau de l'objet mixte, afin d'avoir un prototype maintenable, flexible, et réutilisable pour d'autres contexte d'interaction ;
2. La modularité au niveau des modalités de liaison, avec des modules réutilisables pour les dispositifs, langages et compositions en entrée et en sortie, afin d'avoir un prototype maintenable, flexible et de faciliter les micro-itérations ;
3. L'extensibilité de l'outil. Un développeur doit être capable d'ajouter de nouveaux modules pour étendre la boîte à outils à de nouvelles technologies.

Nous présentons les différents composants offerts actuellement par la boîte à outils, et comment il est possible de les assembler pour construire des prototypes d'objets mixtes.

6.1. Composants pour les dispositifs de liaison

Les composants pour les dispositifs de liaison sont basés sur trois types de ressources matérielles : des caméras (*ARVideoInputDevice*), des cartes d'Interface-Z (*MIDIDevice*) (http://www.interface-z.com/), et un kit électronique des Phidgets (*PhidgetInterfaceKitDevice*) (http://www.phidgets.com/). En l'état actuel de la boîte à outils, les dispositifs standards comme l'écran ou les haut-parleurs peuvent être utilisés avec la boîte à outils OP directement via le langage de liaison. En effet, le logiciel disponible sur les ordinateurs standards gère déjà ces dispositifs. Par conséquent, aucun composant OP n'est fourni pour ces dispositifs.

6.1.1. ARVideoInputDevice

Nous avons choisi d'intégrer la partie capture vidéo de l'ARToolKit (http://www.hitl.washington.edu/artoolkit/) à la boîte à outils. Notre choix s'est tourné vers l'ARToolKit pour sa popularité. La boîte à outils propose donc un composant *ARVideoInputDevice* qui capture l'entrée vidéo de la caméra. La sortie de ce composant est consituée des images de la caméra. Le constructeur de ce composant est le suivant :

```
ARVideoInputDevice(char* pName);
```

Son paramètre *pName* est une chaîne de caractères qui représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).

Pour connecter ce composant, OP fournit plusieurs signaux, qui sont :

```
void updated(QImage* pImage);
void updated(ARUint8* pImage, QTime pTime);
```

Le premier paramètre de ces signaux est une référence sur l'image captée par la caméra. Le second paramètre du deuxième signal est une estampille temporelle. D'autres signaux et slots existent, pour le fonctionnement interne du composant, ou pour la connexion avec l'interface graphique optionnelle. Ils ne sont pas utilisés pour prototyper un objet mixte.

Par exemple, pour prototyper un objet avec une caméra, l'utilisateur d'OP peut écrire :

```
ARVideoInputDevice Camera("Camera");
```

Pour connecter ce composant, il devra compléter la ligne suivante :

```
QObject::connect(&Camera, SIGNAL(updated(ARUint8*, QTime)),
...
```

6.1.2. MIDIDevice

Nous avons choisi d'intégrer dans la boîte à outils OP des dispositifs d'Interface-Z (http://www.interface-z.com/), connaissant leur popularité dans la communauté artistique. Nous avons donc construit un composant logiciel qui encapsule le traitement des messages MIDI et qui permet de capturer/matérialiser des données de/vers des capteurs/effecteurs branchés sur les cartes matérielles Interface-Z. La Figure 189 présente une carte Interface-Z de capteurs tandis que la Figure 190 montre une carte d'effecteurs. La sortie ou l'entrée MIDI de ces cartes doit être branchée

à une interface MIDI/USB (non représentée aux Figure 189 et Figure 190) pour communiquer avec l'ordinateur.

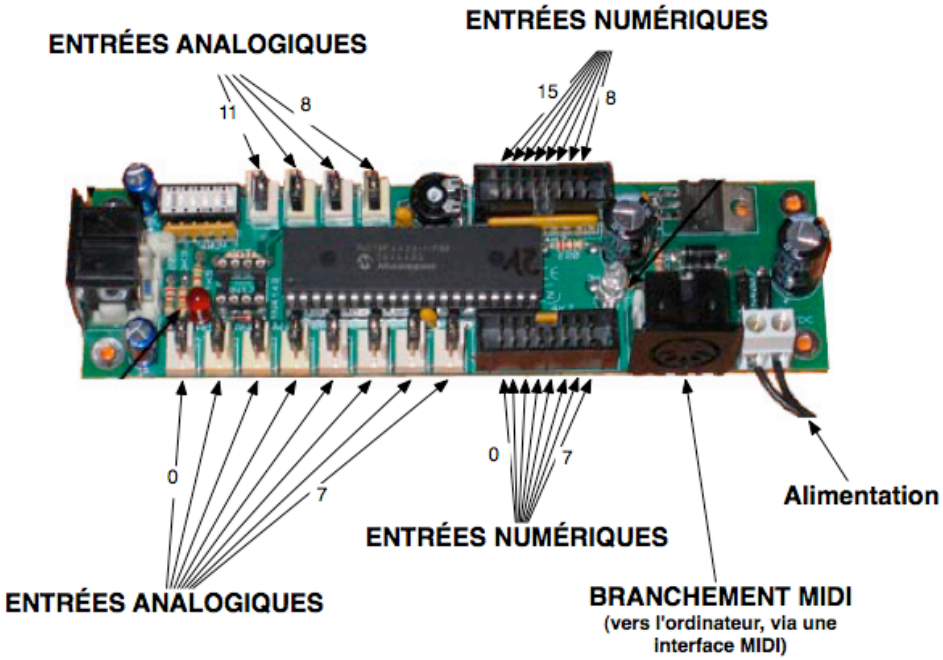


Figure 189 : Carte matérielle d'Interface-Z pour les capteurs. Illustration adaptée de http://www.interface-z.com/.

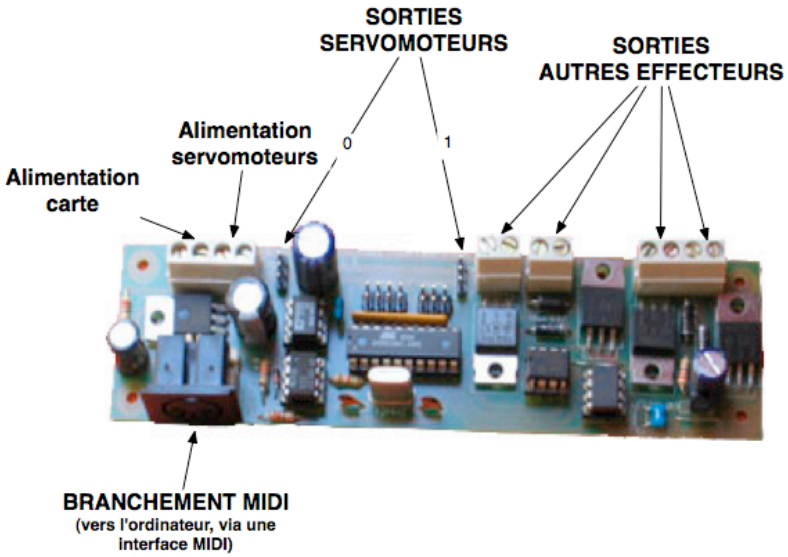


Figure 190 : Carte matérielle d'Interface-Z pour les effecteurs : 2 servomoteurs et 4 autres actionneurs, comme des lampes par exemple. Illustration adaptée de http://www.interface-z.com/.

Sur la carte de la Figure 189 peuvent être branchés 12 capteurs analogiques, comme un capteur de flexion et 16 capteurs numériques comme des boutons ou des interrupteurs. Sur la carte de la Figure 190 peuvent être branchés des effecteurs, comme des servomoteurs ou des lampes par exemple.

Notre composant encapsule le traitement des messages MIDI effectués avec la bibliothèque *CoreMIDI* d'Apple[48]. Ce composant, appelé MIDIDevice, est utilisé indifféremment pour un capteur (entrée) ou un effecteur (sortie). La Figure 191 présente un constructeur qui peut être utilisé pour

[48] Des équivalents à CoreMIDI d'Apple existe pour PC ou Linux (par exemple, http://midiio.sapp.org/).

créer un composant OP pour un dispositif d'entrée ou de sortie. Les paramètres de ce constructeur sont les suivants :

1. Le paramètre *pName* (Figure 191, ligne 1) du constructeur est une chaîne de caractères qui représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pDirection* (Figure 191, ligne 2) du constructeur indique si le dispositif est un dispositif de sortie ou d'entrée, c'est-à-dire s'il est branché à une carte du type de la Figure 189 ou du type de la Figure 190.
3. Le paramètre *pNumber* (Figure 191, ligne 3) du constructeur indique le numéro d'entrée/sortie utilisée. Ce paramètre *pNumber* peut prendre une valeur entière entre 0 et 15. Il correspond à l'emplacement du branchement sur la carte comme le montre la Figure 189 et la Figure 190 (étiquettes sur les flèches désignant les entrées/sorties).
4. Le paramètre *pIsAnalogue* (Figure 191, ligne 4) du constructeur est utilisé s'il s'agit d'un dispositif d'entrée. Il indique dans ce cas si le dispositif est branché sur les entrées analogiques (s'il vaut *vrai*) ou numériques (s'il vaut *faux*).
5. Le paramètre *pResolution* (Figure 191, ligne 5) précise la résolution des données envoyées par la carte Interface-Z. Une carte Interface-Z peut envoyer des données sur 7 ou 12 bits. Avant de donner la valeur 7 ou 12 à ce paramètre, il faut vérifier sur la carte matérielle quelle est la résolution des données qu'elle envoie (http://www.interface-z.com/).

```
1  MIDIDevice(char* pName,
2             LinkingComponent::LinkingComponentDirection pDirection,
3             int pNumber,
4             bool pIsAnalogue,
5             int pResolution);
```

Figure 191 : Pour utiliser un dispositif branché à la carte matérielle de capteurs (Figure 189) ou d'effecteurs (Figure 190).

Pour connecter ce composant, OP fournit plusieurs signaux et slots. Si le dispositif est un dispositif d'entrée, alors les signaux proposés sont :

```
void updated(int pMessage, QTime pTime);
void updated(QVariant pMessage, QTime pTime);
void updated(int pMessage, QTime pTime, QString pName);
```

Pour les trois signaux, le premier paramètre (*pMessage*) est la valeur captée par le dispositif matériel dans le cas d'un dispositif d'entrée ou la valeur envoyée vers le dispositif matériel dans le cas d'un dispositif de sortie. Le second paramètre est une estampille temporelle. Le troisième paramètre (dernier signal) est un identifiant utilisée par l'élément OP de composition (section 2.3).
Si le dispositif est un dispositif de sortie, alors le slot proposé est :

```
void updateOutput(int pValue);
```

Le paramètre est la valeur qui doit être envoyée vers le dispositif matériel.
D'autres signaux et slots existent, pour le fonctionnement interne du composant, ou pour la connexion avec l'interface graphique optionnelle. Ils ne sont pas utilisés pour prototyper un objet mixte.

Par exemple, pour prototyper l'objet de la Figure 186 avec un capteur de flexion d'interface-Z présenté Figure 192, nous déclarons un composant *MIDIDevice* appelé *BendSensor* :

```
int Index = 0;
```
(entouré en rouge sur la Figure 192)

```
bool IsAnalogue = true;
```
(le capteur est analogique)

```
int Resolution = 7;
```
(le capteur envoie des messages en 7 bits donc les valeurs sont dans l'intervalle [0, 127].)

```
MIDIDevice BendSensor("BendSensor", LinkingComponent::IN, Index, IsAnalogue, Resolution);
```

Pour connecter ce composant, l'utilisateur d'OP devra compléter la ligne suivante :

```
QObject::connect(&BendSensor, SIGNAL(updated(int, QTime)), ...
```

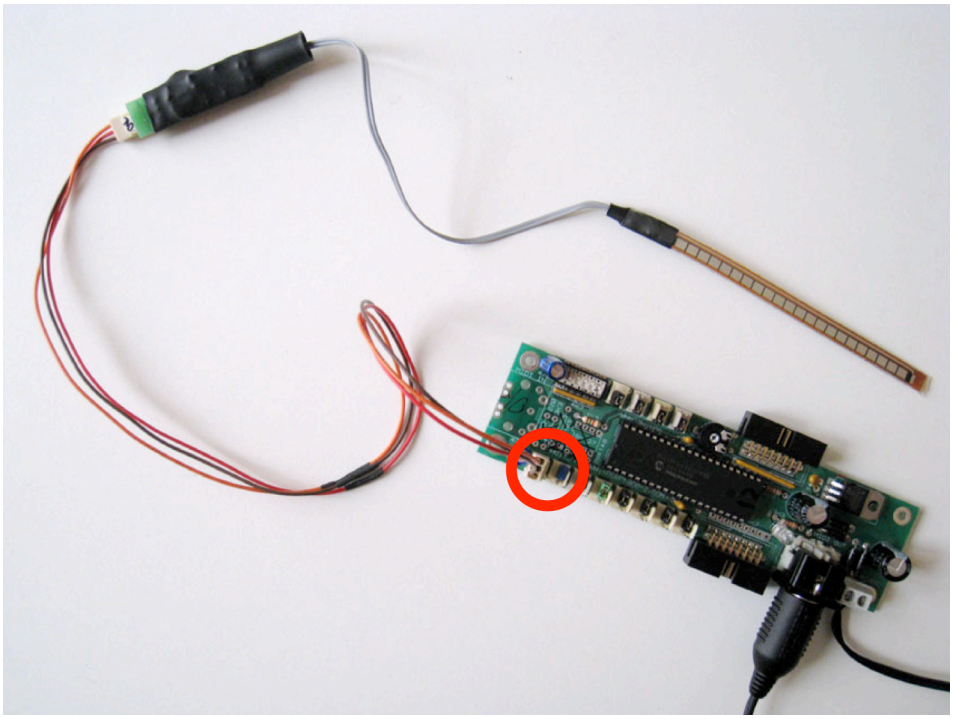


Figure 192 : Exemple de prototypage avec un capteur de flexion interface-Z : configuration matérielle pouvant correspondre au code *MIDIDevice BendSensor("BendSensor", LinkingComponent::IN, Index, isAnalogue, Resolution);* avec Index = 0 (place du branchement, entouré en rouge), isAnalogue = true et Resolution = 7.

6.1.3. PhidgetInterfaceKitDevice

Nous avons choisi d'intégrer à la boîte à outils des dispositifs Phidgets (http://www.phidgets.com/), connaissant leur popularité dans la communauté IHM. Nous avons donc construit un composant OP qui encapsule les fonctions fournit par l'API des Phidgets et qui permet de capturer/matérialiser des données de/vers des capteurs/effecteurs branchés sur un Phidget Interface Kit 8/8/8 (Figure 193).

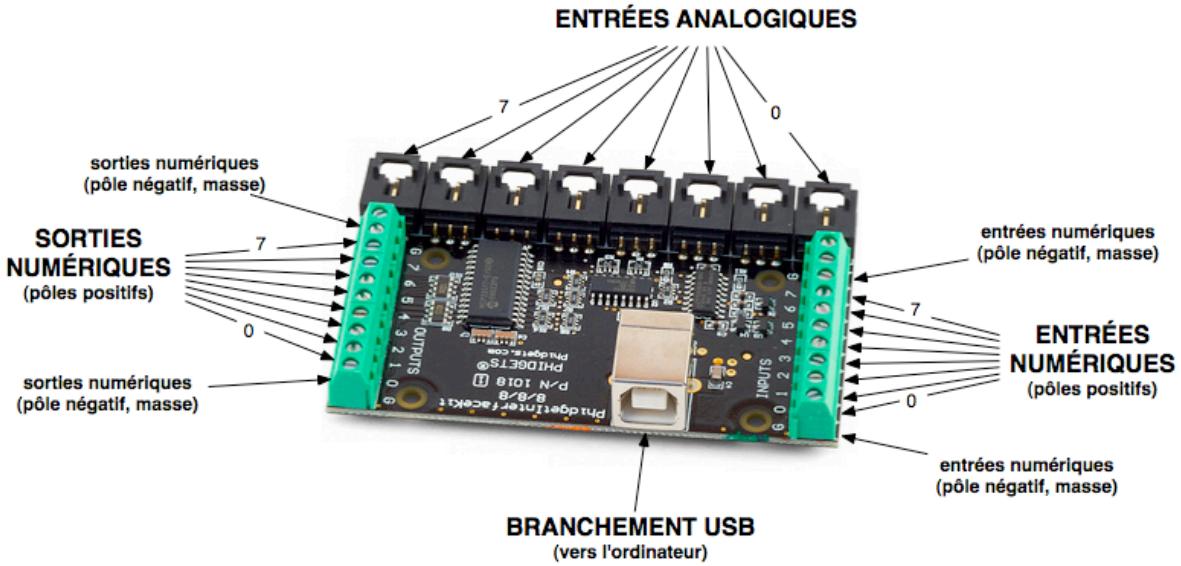


Figure 193 : Phidget Interface Kit 8/8/8. Illustration adaptée de http://www.phidgets.com/.

Sur la carte matérielle de la Figure 193 peuvent être branchés simultanément :

- 8 capteurs analogiques mesurant des grandeurs physiques continues, comme des capteurs de température, de force, etc. (Figure 194),

- 8 capteurs numériques comme des boutons, des interrupteurs, etc. (Figure 195),
- 8 effecteurs numériques comme des diodes électroluminescentes (Figure 196).



Figure 194 : Exemples de capteurs analogiques des Phidgets : (de haut en bas et de gauche à droite) un capteur de toucher, une glissière, un capteur de luminosité, un capteur de force. Illustration extraite de http://www.phidgets.com/.



Figure 195 : Exemples de capteurs numériques des Phidgets : des interrupteurs. Illustration extraite de http://www.phidgets.com/.



Figure 196 : Exemples d'effecteurs numériques : des diodes électroluminescentes. Illustration extraite de http://www.phidgets.com/.

Notre composant OP encapsule les fonctions fournies par l'API des Phidgets. Ce composant, appelé *PhidgetInterfaceKitDevice*, est utilisé indifféremment pour un capteur (entrée) ou un effecteur (sortie). La Figure 197 présente son constructeur.

1. Le paramètre *pName* (Figure 197, ligne 1) du constructeur est une chaîne de caractères qui représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pDirection* (Figure 197, ligne 2) du constructeur indique si le dispositif est un dispositif de sortie ou d'entrée, c'est-à-dire s'il est branché sur les entrées (analogiques ou numériques) ou sur les sorties numériques (Figure 193).
3. Le paramètre *pNumber* (Figure 197, ligne 3) du constructeur indique le numéro d'entrée/sortie utilisée. Ce paramètre *pNumber* peut prendre une valeur entière entre 0 et 7. Il correspond à l'emplacement du branchement sur la carte comme le montre la Figure 193 (étiquettes sur les flèches désignant les entrées/sorties).
4. Le paramètre *pIsAnalogue* (Figure 197, ligne 4) du constructeur est utilisé s'il s'agit d'un dispositif d'entrée. Il indique dans ce cas si le dispositif est branché sur les entrées analogiques (s'il vaut *vrai*) ou numériques (s'il vaut *faux*). Par défaut, s'il est omis dans la déclaration du composant, il vaut *faux*. Ainsi il n'est pas utile de le spécifier dans le cas d'une sortie ou d'une entrée numérique.

```
PhidgetInterfaceKitDevice(char* pName,
                          LinkingComponent::LinkingComponentDirection pDirection,
                          int pNumber,
                          bool pIsAnalogue);
```

Figure 197 : Pour utiliser un dispositif branché à l'interface Kit 8/8/8 des Phidgets.

Pour connecter ce composant, OP fournit plusieurs signaux et slots. Si le dispositif est un dispositif d'entrée, alors le signal proposé est :

```
void updated(int pMessage, QTime pTime);
```

Si le dispositif est un dispositif de sortie, alors les slots proposés sont :

```
void UpdateOutput(bool pMessage);
void updateOutput(QVariant pMessage);
```

Les paramètres représentent la même chose que pour le composant OP précédent (la valeur captée ou à envoyée vers le dispositif matériel et une estampille temporelle). D'autres signaux et slots existent, pour le fonctionnement interne du composant, ou pour la connexion avec l'interface graphique optionnelle. Ils ne sont pas utilisés pour prototyper un objet mixte.

Par exemple, pour prototyper un objet avec une glissière physique en entrée (Figure 198), nous déclarons un composant *PhidgetInterfaceKitDevice* appelé *Slider* :

```
int SliderIndex = 1;
```
(emplacement de branchement de la glissière physique sur la Figure 198, en rouge)

```
bool isSliderAnalogue = true;
PhidgetInterfaceKitDevice Slider("Slider", LinkingComponent::IN, SliderIndex, isSliderAnalogue);
```

Pour connecter ce composant, l'utilisateur d'OP devra compléter la ligne suivante :

```
QObject::connect(&Slider, SIGNAL(updated(int, QTime)),...
```

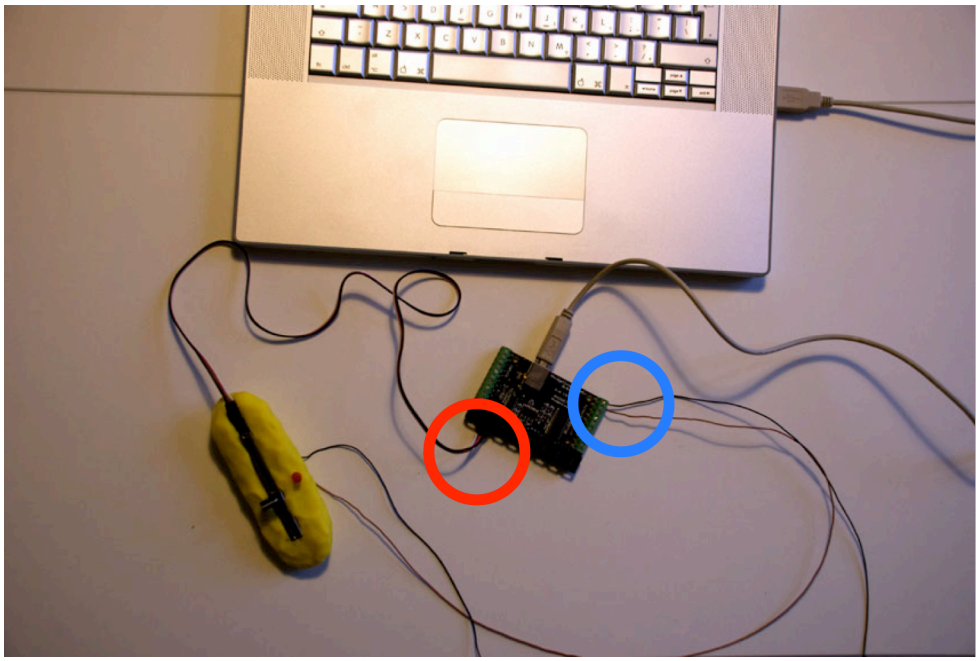


Figure 198 : Exemple de prototypage avec une glissière physique des Phidgets et une diode électroluminescente branchée à une sortie numérique de la même carte : configuration matérielle pouvant correspondre au code *PhidgetInterfaceKitDevice Slider("Slider", LinkingComponent::IN, SliderIndex, isSliderAnalogue); PhidgetInterfaceKitDevice LED("LED", LinkingComponent::IN, LEDIndex, isLEDAnalogue);* avec SliderIndex = 1 (entouré en rouge), LEDIndex = 0 (entouré en bleu), isSliderAnalogue = true, isLEDAnalogue = false.

Pour prototyper ce même objet avec une diode en sortie comme à la Figure 198, nous déclarons un autre composant *PhidgetInterfaceKitDevice* appelé *LED* :

```
int LEDIndex = 0;
```
(emplacement de branchement de la diode sur la Figure 198, en bleu)

```
bool isLEDAnalogue = false;
PhidgetInterfaceKitDevice LED("PhidgetLEDgreen", LinkingComponent::OUT, LEDIndex, isLEDAnalogue);
```
Pour connecter ce composant, l'utilisateur d'OP devra compléter la ligne suivante :
```
QObject::connect( ....................., &LED, SLOT(UpdateOutput(bool)));
```

Nous avons choisi d'intégrer comme trois types de dispositifs, deux boîtes à outils matérielles (Chapitre 6, section 2.1.1 et 2.1.3) très utilisées par deux communautés distinctes : les artistes pour Interface-Z et les chercheurs en IHM pour les Phidgets. Nous avons aussi intégré une boîte à outils logicielle (Chapitre 6, section 2.2.1) qui est très utilisée dans la communauté de Réalité Augmentée. Bien évidemment, d'autres dispositifs peuvent être définis, comme Arduino (Chapitre 6, section 2.1.2).

6.2. Composants pour les langages de liaison

Pour les langages de liaison, la boîte à outils OP fournit aujourd'hui une dizaine de composants.

6.2.1. ARTrackingInputLanguage

Dans la continuité du composant *ARVideoInputDevice*, nous avons choisi d'intégrer la partie traitement d'image et vision par ordinateur de l'ARToolKit (http://www.hitl.washington.edu/artoolkit/) à la boîte à outils. Ce composant permet de suivre un marqueur dans une image vidéo. Le composant propose des sorties selon les informations calculées par l'ARToolKit : par exemple, il est possible de récupérer les coordonnées des sommets du marqueur dans le référentiel de l'image.
La Figure 199 présente le constructeur de ce composant.

1. Le paramètre *pName* (Figure 199, ligne 1) du constructeur est une chaîne de caractères qui représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée),
2. Le paramètre *pThreshold* (Figure 199, ligne 2) indique le valeur de luminance seuil utilisée pour le traitement de l'image.
3. Le paramètre *pMode* (Figure 199, ligne 3) indique le mode de détection du marqueur, qui peut être :
    a. Continu : dans ce cas, l'algorithme utilise l'historique des informations calculées dans l'image précédente.
    b. *One shot* : dans ce cas, l'algorithme n'utilise que les informations dans l'image courante pour trouver le marqueur.
4. Le paramètre *pPattName* (Figure 199, ligne 4) indique le nom du fichier qui contient le modèle de marqueur.

```
1  ARTrackingInputLanguage(char* pName,
2                          int pThreshold,
3                          int pMode,
4                          char* pPattName);
```

Figure 199: Pour utiliser la reconnaissance des marqueurs de l'ARToolKit.

Pour connecter ce composant, OP fournit un signal et un slot. Pour connecter un dispositif d'entrée à ce composant, alors le slot proposé est :

```
void update(ARUint8* pImage, QTime pTime);
```

Le premier paramètre est un pointeur sur l'image à traiter reçue en entrée, le second est une estampille temporelle.
Pour connecter ce composant à un autre composant, comme à une propriété numérique, le signal proposé est :

```
void updated(ARMarkerInfo pMarkerInfo, double pPattTrans[3][4], QTime pTime);
```

Le premier paramètre *pMarkerInfo* indique l'identification du marqueur, le second est une matrice contenant la position et l'orientation du marqueur par rapport à la caméra, enfin *pTime* est une estampille temporelle.

6.2.2. OrientationInputLanguage

Ce composant calcule une orientation à partir d'un couple de deux valeurs. Ce couple peut être par exemple issu de deux composants dispositifs de liaison combinés entre eux via un composant de composition (section 2.3). La résolution de l'orientation fournie est une propriété du composant que l'utilisateur d'OP peut ajuster. Par exemple, si nous souhaitons connaître si l'objet est vers le haut ou le bas alors la résolution est 2. Au contraire, si nous souhaitons connaître si l'objet est vers le haut, le bas, la gauche ou la droite, alors la résolution vaut 4.
La Figure 200 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 200, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pPrecisionLevel* (Figure 200, ligne 2) permet de fixer la résolution de l'orientation.

```
1  OrientationInputLanguage(char* pName,
2                           int pPrecisionLevel);
```

Figure 200 : Constructeur pour créer un *OrientationInputLanguage*.

Pour connecter ce composant, OP fournit un signal et un slot. Pour connecter un dispositif d'entrée à ce composant, alors le slot proposé est :

```
void update(QVector<int> pMessage, QTime pTime);
```

Le premier paramètre est le couple de deux valeurs entières reçu en entrée, le second est une estampille temporelle.
Pour connecter ce composant à un autre composant, comme à une propriété numérique, les signaux proposés sont :

```
void updated(int pMessage, QTime pTime);
void updated(QVariant pMessage, QTime pTime);
```

Le premier paramètre indique la valeur entière renvoyée par le composant, le second est une estampille temporelle.

6.2.3. IdentityInputLanguage

Ce composant ne déforme pas les valeurs d'entrée reçues d'un capteur. Une propriété du composant (*isOpposite*, Figure 201) permet à l'utilisateur d'OP d'opposer les valeurs de sorties. Par exemple, si un capteur fournit des valeurs en entrée entre 0 et 999, alors les valeurs de sortie du composant sont entre 999 et 0. La Figure 201 présente graphiquement la transformation effectuée par le composant.

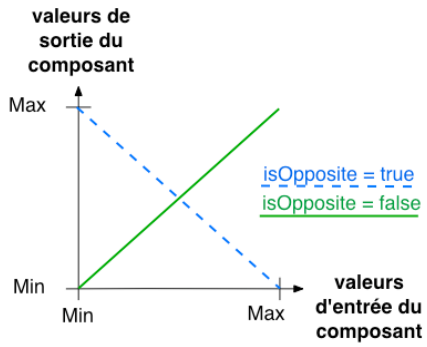


Figure 201 : Transformation effectuée par le composant *IdentityInputLanguage*.

La Figure 202 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 202, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pIsOpposite* (Figure 202 ligne 2) indique si le composant renvoie les valeurs dans leur sens originel (Figure 201 en pointillés bleus) ou dans l'autre sens (Figure 201 en ligne pleine verte).
3. Le paramètre *pMin* (Figure 202 ligne 3 et Figure 201) est un paramètre du constructeur qui permet d'indiquer au composant la valeur minimum envoyée (par le dispositif) en entrée.
4. Le paramètre *pMax* (Figure 202 ligne 4 et Figure 201) est un paramètre du constructeur qui permet d'indiquer au composant la valeur maximum envoyée (par le dispositif) en entrée.

```
1  IdentityInputLanguage(char* pName,
2                        bool pIsOpposite,
3                        int pHardMin,
4                        int pHardMax);
```

Figure 202 : Constructeur pour créer un *IdentityInputLanguage*.

Pour connecter ce composant, OP fournit un signal et un slot. Pour connecter un dispositif d'entrée à ce composant, alors le slot proposé est :

```
void update(int pMessage, QTime pTime);
```

Le premier paramètre est la valeur entière reçue en entrée, le second est une estampille temporelle. Pour connecter ce composant à un autre composant, comme à une propriété numérique, les signaux proposés sont :

```
void updated(int pMessage, QTime pTime);
void updated(QVariant pMessage, QTime pTime);
```

Pour les deux signaux, le premier paramètre indique la valeur entière renvoyée par le composant, le second est une estampille temporelle.

6.2.4. RampInputLanguage

Ce composant généralise le composant précédent et implémente une déformation des valeurs d'entrée en suivant la fonction « rampe » (Figure 203). Les propriétés du composant permettent au d'ajuster des valeurs caractéristiques de la transformation : *IsOpposite*, *LowerThreshold*, *UpperThreshold* (Figure 203). Il est également possible d'indiquer *Min* et *Max* (Figure 203).

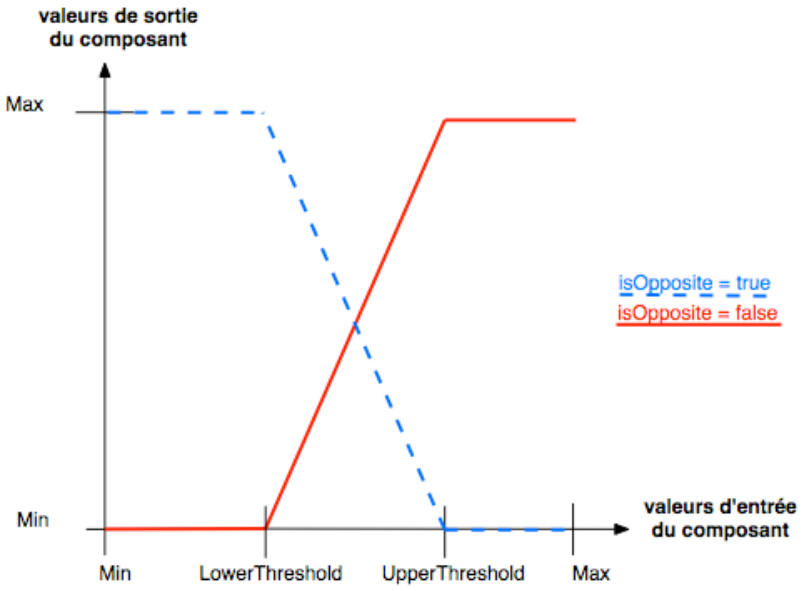


Figure 203 : Transformation effectuée par le composant *RampInputLanguage*. Si *LowerThreshold* vaut *Min* et *UpperThreshold* vaut *Max*, alors nous retrouvons la même transformation que l'identité (Figure 201).

Le constructeur correspondant à ce composant est décrit à la Figure 204. Sa description complète, proche du composant précédent, est fournie en Annexe D.

```
RampInputLanguage(char* pName,
                  bool pIsOpposite,
                  int hardMin,
                  int pLowerThreshold,
                  int pUpperThreshold,
                  int hardMax);
```

Figure 204 : Constructeur pour créer un RampInputLanguage.

6.2.5. ThresholdInputLanguage

Ce composant propose une sortie booléenne vraie (resp. fausse) si la valeur entière en entrée est au-dessus d'un seuil, et fausse (resp. vraie) dans le cas contraire (Figure 205). Les propriétés de ce composant sont *threshold* et *isTrueAbove* (Figure 205).

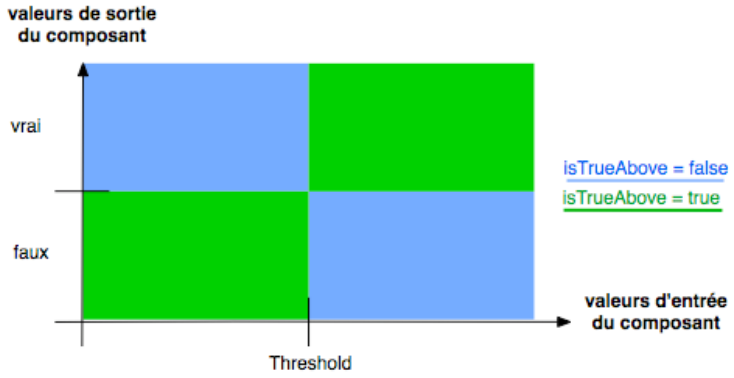


Figure 205 : Transformation effectuée par le composant ThresholdInputLanguage.

La Figure 206 présente le constructeur de ce composant. Sa description complète est fournie en Annexe D.

```
ThresholdInputLanguage(char* pName,
                       int pThreshold,
                       bool pIsTrueAbove);
```

Figure 206 : Pour créer un langage de liaison « seuil ».

6.2.6. DelayLanguage

Indifféremment composant d'une modalité de liaison en entrée ou en sortie (*pDirection* à la Figure 207), ce composant implémente une attente : il ne délivre en sortie sa valeur d'entrée qu'après un certain temps. Cette durée (*pDelay* à la Figure 207)) est une propriété du composant et est ajustable par l'utilisateur d'OP. Ce composant est complètement décrit en Annexe D.

```
DelayLanguage(char* pName,
              LinkingComponent::LinkingComponentDirection pDirection,
              int pDelay);
```

Figure 207 : Pour créer un langage de liaison « délai ».

6.2.7. RepeatLanguage

Comme le composant précédent, ce composant peut être utilisé pour une modalité de liaison en entrée ou en sortie (*pDirection* à la Figure 208). Ce composant répète sa valeur d'entrée plusieurs fois à intervalle régulier. Les propriétés de ce composant (nombre de répétitions, intervalle entre les répétitions) peuvent être ajustées par l'utilisateur d'OP. Ce composant est complètement décrit en Annexe D.

```
1  RepeatLanguage(char* pName,
2                 LinkingComponent::LinkingComponentDirection pDirection,
3                 int pRepeat,
4                 float pInterval);
```

Figure 208 : Pour créer un langage de liaison « répétition ».

6.2.8. ShortDisplayOutputLanguage

Ce composant affiche une image ou un texte fourni en entrée, pendant un certain laps de temps. Cette durée d'affichage (*interval*) est une propriété ajustable du composant, ainsi que le type de ce qui est affiché (texte, image). La Figure 209 présente l'exemple de l'utilisation de ce composant dans le cas de l'outil de lecture d'ORBIS (Chapitre 5, section 3.1.2.1.2.1). Lors de la conception, nous avons essayé parmi d'autres alternatives l'affichage de l'image de la Figure 209 pendant 1 seconde. Ce composant est complètement décrit en Annexe D.



Figure 209 : Exemple d'affichage d'une image par le composant *ShortDisplayOutputLanguage* dans le cas de l'outil de lecture pour ORBIS. (Chapitre 5, section 3.1.2.1.2.1).

6.2.9. SlideshowOutputLanguage

Ce composant affiche une liste d'images, les unes après les autres. Le constructeur proposé est le suivant :

```
SlideshowOutputLanguage(char* pName);
```

Son paramètre *pName* permet de fixer quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
Pour connecter ce composant, OP fournit un signal et des slots. Pour l'entrée, les slots proposés sont :

```
void updatePhoto(QImage *pMessage);
void updatePhoto(QImage *pMessage, int pSide);
```

Le premier paramètre est l'image reçue en entrée, le second (optionnel) permet d'indiquer le côté de l'image qui sera affiché en haut de l'écran.
Pour connecter ce composant à un autre composant, le signal proposé est :

```
void PhotoUpdated(QImage *pMessage);
```

Le paramètre indique ce qui a été affiché par le composant.

6.2.10. BeepOutputLanguage

Ce composant correspond à une fonction impulsion, déclenchée quand le composant est activé via son slot. Il déclenche un bip sur un haut-parleur (dispositif standard) s'il est utilisé avec un fichier

son, ou un clignotement s'il est utilisé connecté à une diode électroluminescente. Ce composant est décrit complètement en Annexe D.

Nous venons d'introduire un ensemble de composants OP correspondant à des langages de liaison. Ces composants sont complètement décrits en Annexe D. Certains de ces composants sont dédiés à des modalités de liaison en entrée, d'autres en sortie et enfin comme *RepeatLanguage* en entrée/sortie. Selon le modèle d'interaction mixte, ils peuvent être connectés à des composants OP dispositifs (section 2.1) ou à des propriétés numériques (section 2.4). De plus, pour définir une modalité de liaison avec un langage élaboré, les composants langages peuvent être connectés en « série » au sein d'une même modalité de liaison. La sortie d'un langage peut être connectée à l'entrée d'un autre. Par exemple pour faire clignoter plusieurs fois une diode électroluminescente, nous connectons un *RepeatLanguage* à un *BeepOutputLanguage*, puis à un *PhidgetInterfaceKitDevice*. Les composants dispositifs et langages constitutifs d'une modalité de liaison étant introduits, nous présentons maintenant le composant de composition de modalités de liaison.

6.3. Composant de composition de modalités de liaison

Pour composer les modalités de liaison, la boîte à outils OP fournit pour le moment un composant *Complementarity* qui permet de combiner de façon complémentaire les données venant de deux autres composants, dispositifs ou langages de liaison. Ce composant est directement inspiré des composants ICARE [Bouchet et al., 2004] pour la fusion (sur critère temporel) des modalités d'interaction dans le cadre des applications multimodales. La Figure 210 présente le constructeur de ce composant.

```
1  Complementarity(char* pName,
2                  int pNbOfConnectedComponents,
3                  int pDeltaT,
4                  AbstractionLevel pLevel);
```

Figure 210 : Pour créer une composition entre deux modalités de liaison.

1. Le premier paramètre du constructeur est *pName* (Figure 210, ligne 1) et permet de fixer quelle chaîne de caractères représente le nom du composant (propriété QObject), tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le second paramètre, *pNbOfConnectedComponents* (Figure 210, ligne 2), permet de fixer le nombre de composants qui sont composés.
3. Le paramètre *DeltaT* (Figure 210, ligne 3), permet d'indiquer la fenêtre temporelle de composition (propriété de la classe mère Composition).
4. Le paramètre *pLevel* (Figure 210, ligne 4) indique quant à lui à quel niveau d'abstraction s'applique cette complémentarité : dispositif ou langage (*pLevel* est aussi une propriété de la classe mère *Composition* (section 1.2)).

Pour connecter ce composant, OP fournit des slots et un signal. Actuellement, les slots proposés sont les suivants :

```
void update1(int pMessage, QTime pTime);
void update2(int pMessage, QTime pTime);
```

Ces deux slots sont connectés aux signaux des deux composants (dispositifs ou langages) à composer. Leur premier paramètre représente la valeur à composer, le second est une estampille temporelle, essentielle pour la fusion sur critère temporel.
Le signal proposé est le suivant :

```
void updated(QVector<int> pMessage, QTime pTime);
```

Le premier paramètre est l'amalgame des premiers paramètres des slots *update1* et *update2*. Le second est leur nouvelle estampille temporelle. Ce signal est émis seulement si les deux candidats (message, time) venant de *update1* et *update2* ont été sélectionnés par l'algorithme.

Par exemple, pour combiner les valeurs sortant des deux accéléromètres de l'objet de la tâche dans ORBIS (Chapitre 5 section 3.1.2.1.1 et Chapitre 8 section 2.2.1.1), nous utilisons le composant *Complementarity*.

6.4. Composant pour les propriétés numériques

Pour les propriétés numériques d'un objet mixte, la boîte à outils OP fournit un composant *DigitalProperty* qui permet de composer un objet mixte avec n'importe quel type de propriété numérique, grâce au type QVariant de Qt (http://trolltech.com/products/qt/).
Pour créer une propriété numérique, nous proposons deux constructeurs (Figure 211 et Figure 212) :

1. Le premier paramètre du constructeur est *pName* (Figure 211 et Figure 212, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pType* (Figure 211 et Figure 212, ligne 2)

```
1  DigitalProperty(char* pName,
2                  char* pType);
```

Figure 211 : Pour créer un composant DigitalProperty, en spécifiant le type de la propriété avec une chaîne de caractères.

```
1  DigitalProperty(char* pName,
2                  QVariant::Type);
```

Figure 212 : Pour créer un composant DigitalProperty, en spécifiant le type de la propriété avec un type QVariant.

Pour connecter ce composant à des modalités de liaison, OP fournit des signaux et un slot. Pour l'entrée, le slot proposé est :

```
void updateProperty(QVariant pProperty);
```

Le paramètre contient la valeur reçue en entrée.
Pour connecter ce composant à des modalités de sorties ou à une application, les signaux proposés sont :

```
void PropertyUpdated(QVariant pProperty);
void PropertyUpdated(QVariant* pProperty);
void updated(int pProperty, QTime pTime);
void updated(bool pProperty, QTime pTime);
```

Les premiers paramètres indiquent la valeur renvoyée par le composant. Le second paramètre, s'il est présent, fournit une estampille temporelle.

Nous venons de présenter les composants à la disposition des utilisateurs d'OP. Nous illustrons maintenant l'utilisation de ces composants sur un exemple, un objet sensible à la lumière.

7. Exemple d'utilisation : objet sensible à la lumière

Nous présentons un exemple d'objet simple qui a été prototypé afin d'illustrer comment utiliser la boîte à outils et ses composants. La Figure 213 présente cet objet qui est sensible à la lumière et qui peut être manipulé par l'utilisateur. L'objet comprend un capteur de luminosité et une diode électroluminescente à sa surface. À chaque fois que l'utilisateur approche l'objet d'une source lumineuse au-delà d'un certain seuil, la diode clignote une fois. S'il éloigne l'objet de la source lumineuse en deçà du seuil, alors la diode clignote une nouvelle fois.

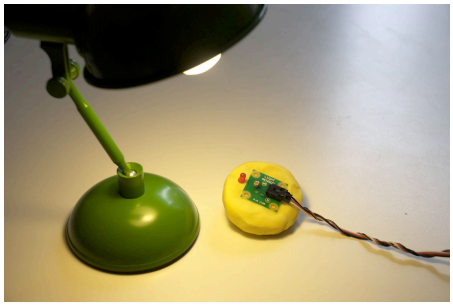


Figure 213 : Exemple d'un prototype d'objet sensible à la lumière.

La Figure 214 présente la description de cet objet mixte selon le modèle d'interaction mixte présenté au Chapitre 4. Un capteur de luminosité (dispositif de liaison en entrée) capture le niveau d'exposition de l'objet à la lumière (propriété physique de l'objet). Si ce niveau est au-dessus d'un certain seuil (langage de liaison en entrée), alors la propriété numérique « est exposé » est vraie. Si le niveau de luminosité descend sous le seuil, alors la propriété numérique « est exposé » est fausse. Chaque fois que l'état de la propriété numérique change, le langage de liaison en sortie est activé. Ce langage entraîne le clignotement de la diode.

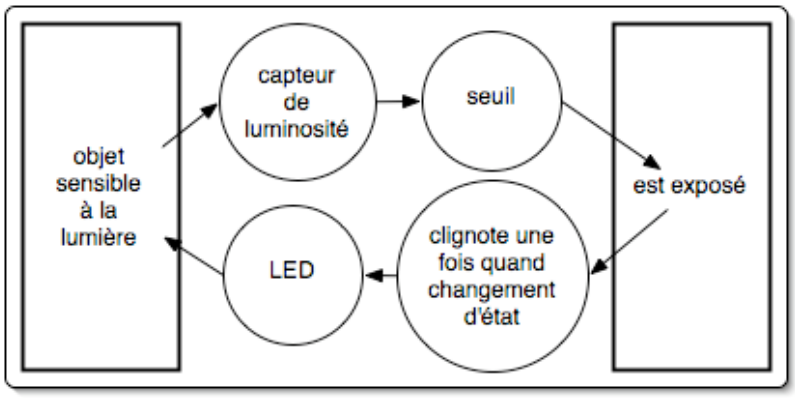


Figure 214 : Description de l'objet sensible à la lumière de la Figure 213, selon le modèle d'interaction mixte.

Nous présentons maintenant comment cet objet peut être prototypé avec OP.

7.1. Prototypage de l'objet

En utilisant les composants, un utilisateur d'OP peut réaliser une ébauche ou prototype de l'objet de la Figure 213 en quelques lignes de code. Dans le code de la Figure 215, nous utilisons :

- (Ligne 3) Un capteur des Phidgets branché sur une entrée analogique d'un Interface Kit 8/8/8. Nous fixons les propriétés de ce composant : son nom est « *lightSensor* », sa direction est « *IN* » (entrée), il est branché à la première entrée de la carte physique électronique des Phidgets (numéro 0), et c'est un capteur analogique (au contraire d'un interrupteur, par exemple).
- (Ligne 7) Un seuil qui renvoie vrai si sa valeur d'entrée est au-dessus de 100, conformément à la description de l'objet présentée Figure 214.
- (Ligne 9) Une propriété numérique « *isExposed* » (« est exposé ») de type booléen.
- (Ligne 12) Un composant *BeepOutputLanguage*, sans fichier son.
- (Ligne 15) Une diode électroluminescente, connectée à la première sortie numérique de la carte physique des Phidgets.

De la ligne 17 à la ligne 24, nous connectons ces cinq composants les uns à la suite des autres, de telle façon que la sortie de l'un fournisse l'entrée du suivant.

```
1  int indexOnBoard = 0;
2  bool isAnalogue = true;
3  PhidgetInterfaceKitDevice lightSensor("lightSensor", LinkingComponent::IN, indexOnBoard, isAnalogue);
4
5  int threshold = 100;
6  bool isTrueAbove = true;
7  ThresholdInputLanguage thresholdIL("thresholdIL", threshold, isTrueAbove);
8
9  DigitalProperty isExposed("isExposed", QVariant::Bool);
10
11 char* soundFile = "";
12 BeepOutputLanguage beep("beep", soundFile);
13
14 isAnalogue = false;
15 PhidgetInterfaceKitDevice led("led", LinkingComponent::OUT, indexOnBoard, isAnalogue);
16
17 QObject::connect(&lightSensor, SIGNAL(updated(int, QTime)),
18                  &thresholdIL, SLOT(update(int, QTime)));
19 QObject::connect(&thresholdIL, SIGNAL(updated(QVariant, QTime)),
20                  &isExposed, SLOT(updateProperty(QVariant)));
21 QObject::connect(&isExposed, SIGNAL(PropertyUpdated(QVariant)),
22                  &beep, SLOT(update(void)));
23 QObject::connect(&beep, SIGNAL(updated(bool)),
24                  &led, SLOT(UpdateOutput(bool)));
```

Figure 215: Code utilisant la boîte à outils OP pour prototyper l'objet de la Figure 213.

Pour cet objet prototypé, nous construisons maintenant une interface graphique pour le contrôler via l'ordinateur.

7.2. Interface graphique

Pour pouvoir interagir avec les différents niveaux d'abstraction de l'objet via l'ordinateur, nous construisons une interface graphique (Figure 216) : De la ligne 25 à la ligne 39, nous ajoutons les composants de l'objet dans des vecteurs. De la ligne 41 à la ligne 50, nous créons l'interface avec ces composants. La ligne 51 donne un nom à la fenêtre et la ligne 52 l'affiche.

```
25
26 QVector<Device*> pInDevice;
27 pInDevice.push_back(&lightSensor);
28
29 QVector<Language*> pIL;
30 pIL.push_back(&thresholdIL);
31
32 QVector<DigitalProperty*> pDP;
33 pDP.push_back(&isExposed);
34
35 QVector<Language*> pOL;
36 pOL.push_back(&beep);
37
38 QVector<Device*> pOutDevice;
39 pOutDevice.push_back(&led);
40
41 MainWidget LightSensitiveObjectWidget(0,
42                                       pInDevice,
43                                       pCompositionEmpty,
44                                       pIL,
45                                       pCompositionEmpty,
46                                       pDP,
47                                       pCompositionEmpty,
48                                       pOL,
49                                       pCompositionEmpty,
50                                       pOutDevice);
51 LightSensitiveObjectWidget.setWindowTitle("Light Sensitive Object");
52 LightSensitiveObjectWidget.show();
```

Figure 216 : Code utilisant l'outil de construction d'interface graphique associé à l'objet codé à la Figure 215. L'interface graphique générée est présentée Figure 217.

Le résultat de l'affichage est l'interface présentée à la Figure 217.

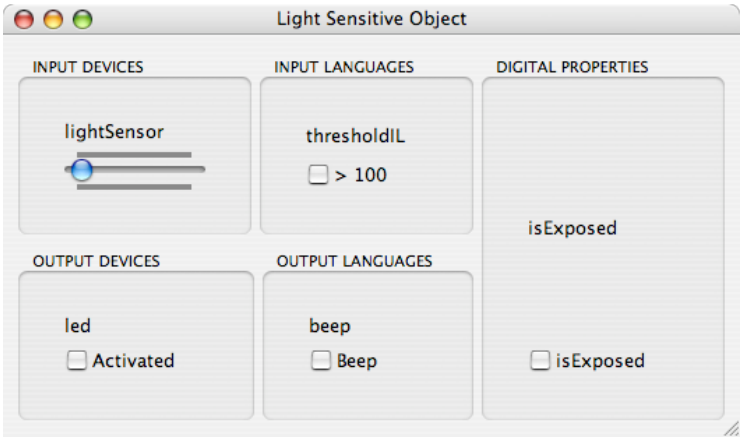


Figure 217 : Interface graphique pour la mise au point de l'objet présenté à la Figure 213 et prototypé à la Figure 215. Le code générant cette interface est présenté à la Figure 216.

Nous présentons maintenant comment un utilisateur d'OP peut faire des modifications pendant la conception dans cette ébauche d'objet.

7.3. Exemples de modifications

Nous présentons des exemples de modifications qu'il est possible de faire pendant la conception à partir de ce prototype. La Figure 218 montre comment faire clignoter la diode quatre fois et la Figure 219 comment changer le capteur des Phidgets pour un capteur MIDI d'Interface-Z.

```
int repeatsNb = 4;
float interval = 0.5;
RepeatLanguage repeat("repeat", LinkingComponent::OUT, repeatsNb, interval);
QObject::connect(&isExposed, SIGNAL(PropertyUpdated(QVariant)),
                 &repeat, SLOT(update(QVariant)));
QObject::connect(&repeat, SIGNAL(updated(bool)),
                 &beep, SLOT(update(void)));
```

Figure 218 : Faire clignoter la diode électroluminescente quatre fois, à partir du code de la Figure 215.

```
int indexOnBoard = 0;
bool isAnalogue = true;
int resolution = 7;
MIDIDevice lightSensor("LightSensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);
```

Figure 219 : Utiliser un capteur de luminosité d'Interface-Z, à la place du capteur des Phidgets de la Figure 215.

À la Figure 218, nous ajoutons un composant *RepeatLanguage* et nous le connectons à ses composants voisins. Ceci correspond à un cas de composition de langage en série comme expliqué à la fin de la section 2.2. À la Figure 219, nous changeons le composant Phidget pour un composant MIDI Interface-Z. Pour cette modification de type de dispositif, il n'est pas nécessaire de reconnecter les composants entre eux, car la modification n'a aucune incidence sur les connexions.

Nous venons de présenter un exemple d'utilisation de la boîte à outils OP pour le prototypage d'un objet simple sensible à la lumière. Nous avons vu notamment comment il est possible de faire rapidement des modifications dans une première ébauche d'objet. Nous présentons d'autres exemples d'utilisation de la boîte à outils OP dans le Chapitre 8 consacré à l'évaluation d'OP.

Dans ce chapitre, nous avons présenté et illustré les composants OP qui permettent une transition fluide entre conception intrinsèque du niveau objet et prototypage. Quand les concepteurs travaillent au niveau extrinsèque c'est-à-dire au niveau application, ils ont besoin d'intégrer l'objet

mixte dans une application. L'outil de prototypage OP ne traite pas ce niveau : en effet, nous avons vu dans l'étude des outils existants du Chapitre 6 que le niveau extrinsèque (interaction avec l'application) est déjà traité par des outils existants (par exemple ICON ou la plateforme OpenInterface). Fidèle à notre démarche intégratrice, l'outil OP peut être considéré comme une fabrique d'éléments de base pour d'autres outils existants comme ICON ou OpenInterface. Pour cela, nous expliquons maintenant comment un objet mixte peut être exploité dans une application, que ce soit un autre outil ou un système de réalité mixte particulier.

8. Insérer un objet mixte dans une application

La boîte à outils OP permet de prototyper des objets mixtes qui peuvent être utilisés et adaptés pour différentes applications. Pour qu'un objet soit utilisé dans une application, c'est le composant *DigitalProperty* de l'objet qui sert d'interface avec l'application.

En effet dans le modèle d'interaction mixte, nous avons identifié des ports d'entrée/sortie numériques (section 2.2.6 du Chapitre 4) qui permettent au reste du système interactif d'acquérir/modifier une valeur d'une propriété numérique. Ainsi les slots constituent une implémentation des ports numériques d'entrée et les signaux sont une implémentation des ports numériques de sortie.
Les signaux et les slots d'un composant *DigitalProperty* qui peuvent être connectés au reste de l'application. La valeur d'une propriété numérique peut être modifiée par l'application via le slot du composant *DigitalProperty*, et une propriété numérique peut communiquer avec l'application en émettant un signal quand sa valeur a changé. C'est ce même mécanisme qui sera utilisé, que l'application ait été développée avec Qt ou non.
Une étude auprès des personnels du Laboratoire d'Informatique de Grenoble (LIG) montre que sur 69 réponses, 19 personnes avaient une connaissance de Qt, de débutant à expert (Chapitre 8 sur l'évaluation). Donc, même si Qt est effectivement utilisé, ce n'est évidemment pas la seule solution.
Après avoir expliqué comment insérer un objet dans une application développée avec Qt, nous présentons donc comment insérer un objet dans une application qui n'a pas été développée avec Qt.

8.1. Insérer un objet dans une application développée avec Qt

Si l'application est écrite avec Qt (et que l'on dispose du code source), alors connecter un objet au reste de l'application est simple. Il suffit d'écrire une ligne de code connectant le signal de la propriété numérique au slot voulu de l'application :

connect(DigitalProperty, SIGNAL(propertyUpdated), ApplicationComponent, SLOT(ApplicationSlot))

Pour communiquer de l'application vers l'objet, il suffit d'écrire une ligne de code connectant le signal voulu de l'application au slot de la propriété numérique :

connect(ApplicationComponent, SIGNAL(ApplicationSignal), DigitalProperty, SLOT(updateProperty))

Nous présentons maintenant comment insérer un objet dans une application développée sans Qt, ou dont nous ne disposons pas du code source.

8.2. Insérer un objet dans une application développée sans Qt

Pour connecter un objet mixte avec une application qui n'est pas écrite en Qt, tout en garantissant l'indépendance de l'objet vis-à-vis de l'application, l'utilisateur d'OP doit définir un *QObject* simple

dont le slot appelle la fonction voulue de l'application. Le slot sera ensuite connecté au signal de l'objet mixte. Ce *QObject* à ajouter sert d'interface entre l'objet et l'application.

Par exemple pour insérer l'objet de la Figure 213 dans l'application Google Earth (http://earth.google.com/), nous avons construit un *QObject* simple avec le slot présenté à la Figure 220. Pour faire tourner le globe terrestre dans Google Earth dans le plan horizontal, il faut appuyer sur la touche 'd' du clavier. Ainsi quand le slot reçoit la valeur « vrai », il simule un appui sur la touche 'd' du clavier (Figure 220). Quand il reçoit la valeur « faux », il simule un relâchement de la touche 'd' (Figure 220). Pour cela, nous utilisons une fonction de la bibliothèque *ApplicationServices* d'Apple qui permet de simuler des évènements clavier. L'équivalent existe pour les autres plateformes.
Si nous connectons ce slot *update* au signal *PropertyUpdated* de la propriété numérique *isExposed* (Figure 215), l'objet sensible à la lumière de la Figure 213 permet alors à l'utilisateur de tourner le globe terrestre à chaque fois que l'objet est proche d'une source lumineuse.

```
void KeySimulationInteractionLanguage::update(QVariant pMessage)
{
    if (pMessage.canConvert(QVariant::Bool))
    {
        if (pMessage.toBool())
        {
            CGPostKeyboardEvent('d', (CGKeyCode)0x02, true);
        }
        else
        {
            CGPostKeyboardEvent('d', (CGKeyCode)0x02, false);
        }
    }
}
```

Figure 220: Slot écrit dans un *QObject* simple servant d'interface entre l'application Google Earth (http://earth.google.com/) et l'objet sensible à la lumière de la Figure 213. L'objet peut ainsi servir à tourner la terre dans Google Earth. CGPostKeyboardEvent est une fonction du ApplicationServices framework d'Apple qui permet de simuler des évènements clavier.

9. Synthèse

Nous avons proposé un outil de prototypage, nommé OP, cohérent avec le modèle d'interaction mixte présenté au Chapitre 4. Nous rappelons que l'objectif de l'outil OP est double : au niveau du prototypage isolé, mais aussi en considérant le prototypage comme un aspect complémentaire de la conception. Nous avons affiné le premier objectif en 4 sous-objectifs au Chapitre 6 section 1. Nous concluons ce chapitre en montrant comment nos objectifs sont atteints par l'outil OP.

En considérant le prototypage de façon isolée, notre outil a été conçu pour permettre l'exploration de la forme et de l'interaction à une résolution plus haute que pour les outils existants comme BOXES (Chapitre 6 section 2.1.6) : l'étendue des possibilités de modalités de liaison à explorer est plus importante avec OP qu'avec BOXES (où les modalités sont équivalentes à celles au clavier et à la souris).

Ensuite, pour augmenter la résolution des premiers prototypes, nous n'avons pas cherché à résoudre un problème technologique local, mais nous avons intégré des technologies nécessitant une expertise particulière, comme des dispositifs d'électroniques (avec les composants *MIDIDevice* et *PhidgetInterfaceKitDevice*) ou des éléments dispositifs et langages de vision par ordinateur pour la réalité augmentée (avec les composants *ARVideoInputDevice* et *ARTrackingInputLangauge*). Ainsi les premiers prototypes peuvent facilement utiliser ces technologies et avoir une plus haute résolution.

Ensuite, pour faciliter une évolution rapide en facilitant les modifications dans le cadre de micro-itérations, nous avons construit des composants élémentaires réutilisables. Nous avons aussi basé

notre outil sur Qt (http://www.qtsoftware.com) et son mécanisme de connexions entre éléments. De plus, nous avons mis à la disposition du développeur une interface graphique pour la mise au point ou la simulation des prototypes.

Enfin, pour intégrer les technologies fournie dans OP, nous avons fait le choix d'intégrer des boîtes à outils existantes comme les Phidgets et l'ARToolKit. Cet aspect montre que la boîte à outils peut être étendue à d'autres technologies. Dans cette direction, nous souhaitons maintenant intégrer Arduino (http://www.arduino.cc/) à notre outil de prototypage.

À plus grande échelle, en repositionnant le prototypage dans la conception, OP fournit des éléments de prototypage pour chaque niveau d'abstraction (non physique) de l'objet mixte (dispositifs et langages de liaison en entrée et sortie, propriétés numériques). OP permet aussi de disposer de modules/« objets mixtes » pouvant prendre part à l'interaction avec une application en tant qu'outil ou objet de la tâche. Il permet ainsi une transition fluide et directe entre les solutions modélisées et leur matérialisation.
Contrairement aux outils existants présentés au Chapitre 6, cet outil permet de prototyper l'interaction à un niveau intermédiaire entre les dispositifs physiques et l'application.

La boîte à outils OP fait aujourd'hui environ 8000 lignes de code et sa documentation est disponible sur http://iihm.imag.fr/coutrix/OP/documentation/.
Parmi les perspectives immédiates de ce travail, outre l'intégration d'Arduino, nous envisageons de développer davantage de composants, par exemple pour proposer d'autres types de compositions des modalités de liaison comme la redondance ou l'équivalence.
Nous sommes également en cours de développement d'un outil annexe pour instrumenter l'évaluation grâce à la génération optionnelle de traces logicielles avec un outil de visualisation de ces traces. Ainsi nous pourrions davantage accompagner les micro-itérations effectuées pendant la conception.
De plus l'insertion de la boîte à outils OP dans un outil de prototypage de l'interaction comme OpenInterface est en cours de réalisation.

Chapitre 8 : Carnet de route et évaluation de la boîte à outils OP

Ce chapitre est consacré au carnet de route de l'élaboration de la boîte à outils OP et à son évaluation : ses apports et ses limitations. Nous présentons d'abord les enjeux et les problématiques liés à l'évaluation des outils pour l'Interaction Homme-Machine. En effet, il est nécessaire de remettre l'évaluation de notre boîte à outils OP dans son contexte, car la difficulté de l'évaluation des outils pour l'Interaction Homme-Machine est reconnue, et les formes d'évaluation proposées dans la littérature ont influencé la façon dont les études et l'évaluation de notre boîte à outils ont été menées.
Après avoir dressé un état de l'art des méthodes d'évaluation, nous présentons ensuite les différentes études et formes d'évaluation de la boîte à outils que nous avons menées.

1. Évaluation des boîtes à outils

Une revue de la littérature permet de constater qu'il n'existe pas de forme d'évaluation consensuelle des outils pour l'interaction-homme machine et beaucoup de propositions existent. Nous étudions d'abord les cadres d'évaluation existants, puis nous analysons des évaluations qui ont été menées.

1.1. Cadres d'évaluation

Parmi les approches existantes, nous trouvons des métriques pour l'évaluation du logiciel et des boîtes à outils. Il existe aussi des critères d'évaluation comme ceux de Klemmer [Klemmer et al., 2004] ou de Myers [Myers, 2000] et les propositions d'évaluation pour éviter le piège de l'utilisabilité de [Olsen, 2007]. Nous les présentons après les métriques.

1.1.1. Métriques

Parmi les métriques existantes pour évaluer la qualité d'une boîte à outils, nous trouvons :

- Le **nombre de lignes de code** qu'un utilisateur doit écrire [Klemmer et al., 2004],
- La **performance** [Klemmer et al., 2004] (latence et consommation des ressources matérielles, comme le processeur),
- La **fiabilité** de la boîte à outils [Klemmer et al., 2004],
- Le **temps** nécessaire pour qu'un utilisateur produise du code [Truong, 2005].

Ces métriques pour l'évaluation du logiciel et des boîtes à outils ont leurs limites. Tout d'abord, elles ne sont pas nécessairement adaptées aux outils de prototypage : en effet, les besoins sont différents entre la production industrielle de logiciel et le prototypage expérimental à petite échelle. Par exemple, si la fiabilité d'un logiciel industriel est importante, ce n'est pas une qualité essentielle pour un outil de prototypage, car les prototypes visent à être manipulés un court instant. Pour pallier cette limitation, [Ballagas et al., 2007] et [Hudson, Mankoff, 2006] ont proposé des métriques plus adaptées au cas d'OP, comme le **nombre d'itérations de conception** effectuées lors de tests. Dans [Ballagas et al., 2007], le nombres d'itérations est défini comme le nombre de cycle « conception → implémentation → analyse » : à chaque fois que les développeurs ont créé un prototype implémentant une partie des fonctionnalités demandées, puis analysé ses problèmes et fait des changements dans leurs choix de conception, les expérimentateurs comptait une itération.

Une autre limite aux métriques reste leur difficulté d'interprétation. Les métriques permettent une évaluation quantitative. Même si une mesure peut être convaincante, elle est difficile à interpréter [Truong, 2005]. En effet, le nombre de lignes de code ou le temps de développement est dépendant de l'expertise des sujets, mais aussi de leur habitude et de leur préférence (subjectivité). Par exemple, le nombre de lignes de code est dépendant de la structure de ce code : en imbriquant les appels de fonctions par exemple, nous pouvons réduire le nombre de lignes de code. Inversement, en choisissant de faire un code modulaire, on augmente le nombre de lignes [Truong, 2005]. Pour pouvoir interpréter ces mesures, il faudrait contrôler le processus de production du prototype, mais ceci ne peut pas être fait sans biaiser les résultats.

1.1.2. Critères de Klemmer

Dans [Klemmer et al., 2004] sont défini des critères pour l'évaluation des boîtes à outils logicielles.

Le premier critère est la **facilité d'utilisation**. Ce critère se décompose en trois dimensions :

- La première dimension est la lisibilité/compréhensibilité des programmes produits par d'autres développeurs avec la boîte à outils.
- La seconde dimension est la compréhensibilité de la boîte à outils pour les développeurs novices.
- La troisième dimension est la facilité d'apprentissage de la boîte à outils.

Le second critère proposé par [Klemmer et al., 2004] est la **réutilisabilité** : une boîte à outils doit fournir des solutions pour des problèmes élémentaires rencontrés souvent. De plus, lorsqu'un développeur rencontre un problème similaire à un problème précédent, la solution fournie par la boîte à outils doit être similaire pour ces deux problèmes.
Le troisième critère est la **constance des codes produits**, que ce soit entre les développeurs ou entre les différents projets pour un seul développeur. Cette constance dans le code influe sur le nombre d'erreurs de conception, la collaboration entre développeurs, ainsi que la maintenabilité du code [Klemmer et al., 2004].

1.1.3. Critères de Myers

Dans [Myers et al., 2000], sont introduits six critères pour évaluer les outils logiciels pour l'Interaction Homme-Machine. Ces critères sont les suivants :

- L'**utilité** : Un outil logiciel doit être utile et aider le développement des parties de l'interface où cela est nécessaire, et seulement là où cela est nécessaire.
- La **difficulté de prise en main** (« seuil ») : Un outil logiciel doit être facile à apprendre (« seuil » bas, facile à franchir). Ce critère rejoint la notion de facilité d'apprentissage de [Klemmer et al., 2004].
- La **puissance** (« plafond ») : Un outil logiciel doit avoir un grand pouvoir d'expression et permettre le développement d'un grand nombre d'interfaces. L'étendue des possibilités définit la puissance de l'outil.
- La **capacité à aiguiller vers de meilleures interfaces** : Un outil logiciel doit être capable d'aiguiller ses utilisateurs vers de meilleures interfaces (vers des solutions de meilleure qualité) et les éloigner des solutions de mauvaise qualité.
- La **prédictibilité** de l'utilisation : Dans le cas de la génération de code, l'utilisateur de l'outil doit être capable de prédire le résultat. Dans [Myers et al., 2000], il est souligné que les outils ne vérifiant pas la prédictibilité ont eu peu de succès.
- L'**importance du problème** qu'il résout : Un outil logiciel doit résoudre un problème nouveau et difficile pour les développeurs. Ainsi un outil qui propose d'aider le développement d'une interaction nouvelle et difficile à réaliser aura du succès.

Dans [Myers et al., 2000], aucun protocole d'évaluation pour ces six critères n'est proposé. Néanmoins nous trouvons dans [Hartmann et al., 2006] et [Ballagas et al., 2007] des propositions d'évaluation de certains de ces critères (section 1.2).

Ces critères introduisent de nouvelles dimensions adaptées aux outils logiciels pour l'Interaction Homme-Machine.
Nous attirons l'attention sur le fait qu'évaluer la facilité d'utilisation pour une boîte à outils encore en phase de recherche n'est peut-être pas le critère d'évaluation le plus important à considérer. Ce point est souligné dans le paragraphe suivant.

1.1.4. Pièges de l'utilisabilité

On trouve dans [Greenberg, Buxton, 2007] une explication de la raison pour laquelle les évaluations effectuées ne sont pas forcément pertinentes. Nous présentons d'abord les faits à la source du

problème, puis leurs conséquences, avant d'exposer les solutions proposées [Greenberg, Buxton, 2007].

1.1.4.1. Faits

Parmi les nombreuses méthodes d'évaluation existantes, comme les études quantitatives, qualitatives, analytiques, informelles ou contextuelles, toutes ne sont pas utilisées dans les mêmes proportions. En effet, l'étude présentée dans [Greenberg, Buxton, 2007] souligne que presque trois quarts des évaluations des travaux publiés à la conférence CHI sont quantitatives. Ils remarquent aussi que la communauté de CHI s'est désintéressée des publications sur de nouvelles formes d'évaluations.
En plus de cette observation des publications à la conférence CHI, les auteurs constatent aussi que les nuances et variations de recherche dans les domaines nouveaux et exploratoires sont plus difficiles à publier et à faire accepter par la communauté. En effet, cette communauté les maîtrise moins bien que les interfaces graphiques par exemple, et n'en discerne pas bien les nuances car elle en est moins spécialiste.

À partir de ce constat, les auteurs en présentent les conséquences.

1.1.4.2. Conséquences

Une idée nouvelle risque d'être écartée à cause de son évaluation, indépendamment de son intérêt [Greenberg, Buxton, 2007]. De plus, la forme d'évaluation quantitative qui prédomine encourage les chercheurs à travailler sur des sujets facilement évaluables, ce qui biaise les travaux et les restreint à un sous-ensemble de possibilités de champs de recherche [Greenberg, Buxton, 2007]. En effet, comme une idée nouvelle est mal maîtrisée, une évaluation de son utilisabilité donnerait probablement de mauvais résultats. Enfin, les évaluations quantitatives découragent la recherche à prendre position et à proposer des solutions subjectives, bien que ce soit des visions de ce type qui permettent d'apporter de nouvelles idées [Greenberg, Buxton, 2007].

Face à ces problèmes, des solutions sont proposées.

1.1.4.3. Solutions

Une évaluation quantitative, pour être convaincante, ne doit pas chercher une situation favorable où l'hypothèse évaluée est vérifiée [Greenberg, Buxton, 2007]. Au contraire, elle doit prendre plus de risque et chercher les limitations autant que les avantages. En effet, les limitations sont le point de départ de nouvelles recherches : par conséquent, même si la confirmation d'une réponse à un problème est intéressante, les nouvelles questions et les nouveaux problèmes que l'évaluation pose sont tout aussi importants.

Il est également important de fournir les éléments pour que la réplication/comparaison puisse être effectuée par d'autres expérimentateurs [Greenberg, Buxton, 2007]. Lié à cette recommandation, nous avons déjà souligné au Chapitre 6 que beaucoup d'outils de la littérature ne sont pas disponibles et donc ne permettent pas de mener des études comparatives par exemple.

Conception centrée utilisateur ne signifie pas *évaluation de l'utilisabilité* [Greenberg, Buxton, 2007]. D'autres éléments entrent en jeu, comme comprendre les requis, considérer les aspects culturels, produire des alternatives de design, etc. Les études ethnographiques et sur le terrain, contrairement aux méthodes répandues d'évaluation, permettent d'étudier les outils dans leur contexte culturel [Greenberg, Buxton, 2007]. Elles sont pourtant très peu utilisées. La validation d'un travail peut prendre la forme d'un *design rationale*[49], de perspectives, de scénarios d'utilisation, de cas d'études,

[49] Rationnel de conception : raisons, logique, justification de la conception : exposer pourquoi les solutions de conception ont été conçues.

de séance de critique constructive participative, d'exposé de ce qu'ils ont appris, des faiblesses des travaux, ou de mise en relation avec d'autres travaux [Greenberg, Buxton, 2007].

Les méthodes utilisées dans les écoles d'art et de design forment des gens créatifs et innovants, qui peuvent s'opposer à la rigueur scientifique de l'Interaction Homme-Machine [Greenberg, Buxton, 2007]. Nous pensons aussi que le domaine a beaucoup à gagner en intégrant artistes et designers et leurs méthodes de conception. Nous trouvons en particulier une approche qui consiste à trouver le bon design parmi l'ensemble des alternatives envisagées pendant la conception. Cette approche s'oppose à l'approche qui consiste à faire des itérations sur une unique idée [Buxton, 2007]. Cette méthode des ébauches, avec celle de la critique constructive collective, peuvent être appliquées en Interaction Homme-Machine.

Nous venons de voir les mises en garde et recommandation soulignées dans [Greenberg, Buxton, 2007] qui souhaitent éviter les évaluations quantitatives de l'utilisabilité, là où elles n'ont pas lieu d'être. Nous présentons maintenant le cadre pour l'évaluation d'outils logiciels [Olsen, 2007], qui s'inscrit dans ce mouvement.

1.1.5. Cadre pour l'évaluation d'outils

Dans [Olsen, 2007], l'auteur propose une méthode détaillée pour évaluer les outils pour l'Interaction Homme-Machine sans tomber dans le piège de l'utilisabilité que nous venons de présenter. Cette méthode se décompose en plusieurs étapes. Tout d'abord, il faut justifier la contribution pour une situation, une tâche et des utilisateurs particuliers. Ensuite, l'auteur propose des méthodes d'évaluation correspondant aux différentes contributions possibles. Enfin, l'auteur rappelle l'importance d'évaluer les lacunes éventuelles d'un outil.

1.1.5.1. Importance

Tout d'abord, l'auteur préconise de justifier **l'importance et l'utilité de la contribution** : les développeurs ne vont pas abandonner un outil avec lequel ils ont l'habitude de travailler pour une amélioration qui n'est pas assez signifiante. Parfois même plus de 100% d'amélioration est nécessaire [Olsen, 2007]. Pourtant, dans certains domaines comme celui du prototypage des objets mixtes, il n'existe pas d'outils consensuels.
Ensuite, l'importance et l'utilité de la contribution doivent être justifiées **pour la population visée**. Il s'agit de justifier l'importance de la tâche réalisable avec l'aide de l'outil pour cette population et **dans la situation particulière qui est prise en compte par l'outil**. De plus, l'auteur préconise d'étudier les différences dans les habitudes de travail qu'impliquent le nouvel outil.

1.1.5.2. Évaluation d'une contribution

Pour ensuite mettre en œuvre une évaluation, l'auteur passe en revue les différentes contributions revendiquées par les outils pour l'Interaction Homme-Machine et propose des méthodes d'évaluation pour chacune. Parmi ces contributions, certains outils pour l'Interaction Homme-Machine permettent de réduire l'effort pour faire des itérations dans l'implémentation d'un système. Cette contribution possible d'un outil est proche des objectifs que nous nous sommes fixés pour OP. Aussi nous la détaillons ci-après.

L'effort nécessaire à fournir pour faire des itérations dans l'implémentation d'un système peut se décomposer en trois dimensions. Ces dimensions sont la **flexibilité**, le **pouvoir expressif**, et la **correspondance expressive**. Ces trois dimensions ne s'évaluent pas de la même façon [Olsen, 2007].

- La flexibilité est la possibilité de faire des changements rapides. Pour l'évaluer, il convient de définir un ensemble de changements à faire, et montrer que ces changements sont significativement plus faciles avec l'outil évalué qu'avec d'autres outils.

- Le pouvoir expressif est la possibilité de réaliser plus en faisant moins d'effort. Pour l'évaluer, il faut montrer que l'outil impose un minimum de choix de conception pour une solution (moins de choix que les autres boîtes à outils). L'effort nécessaire est moindre si le nombre de choix à faire est moindre. Pour cela, un outil peut par exemple éliminer les choix répétitifs en les regroupant. Il peut aussi réduire le nombre de choix à faire en étudiant les dépendances entre les choix pour calculer automatiquement la valeur à choisir pour le dernier choix à faire. Par exemple pour notre outil de prototypage OP, il s'agirait de ne proposer que le sous-ensemble de langages de liaison possible une fois que le concepteur a choisi un dispositif et une propriété numérique.
- La correspondance expressive est la relation entre le problème à résoudre et les moyens d'exprimer les choix de conception. C'est par exemple une contribution de Exemplar [Hartmann et al., 2007] : l'outil vise à permettre au concepteur de spécifier l'interaction avec un capteur en en faisant une démonstration, au lieu d'écrire des lignes de code. Pour tester la correspondance expressive, il faut montrer que la nouvelle forme d'expression correspond davantage au problème pour la situation, la tâche et les utilisateurs visés. Il est possible d'évaluer la correspondance expressive en montrant que, pour la tâche considérée, il est plus facile de corriger les fautes avec un outil plutôt qu'avec un autre. Les sujets doivent localiser et réparer l'erreur. Nous pouvons alors mesurer le temps nécessaire, les erreurs, les difficultés rencontrées, le taux de réussite, afin de comparer des outils entre eux.

Outre cette évaluation adaptée à une contribution possible des outils, les lacunes éventuelles, communes à tous les outils, ne doivent pas être mises de côté [Olsen, 2007].

1.1.5.3. Lacunes éventuelles

Il est rappelé dans [Olsen, 2007] qu'il est nécessaire de se demander quelle est la faiblesse d'un outil pour pouvoir l'évaluer. Ainsi il est possible de contrôler ses limitations, et ne pas revendiquer une contribution trop forte.
Un autre question importante est celle du passage à l'échelle : est-ce que l'outil permet de résoudre un problème non contrôlé et rencontré in situ ? Pour cela il convient d'utiliser l'outil pour résoudre des problèmes réalistes ou de prouver que des problèmes plus complexes n'existent pas.

Nous avons présenté différentes approches pour l'évaluation d'outils logiciels, incluant des métriques pour le logiciel, des critères et méthodes d'évaluation. Au regard de ces approches, nous analysons les évaluations qui ont été menées pour les outils présentés au Chapitre 6, afin d'enrichir nos évaluations d'OP.

1.2. Analyse d'évaluations d'outils

Certains outils, comme Arduino par exemple, n'ont pas donné lieu à des évaluations. Pour Arduino, une forme de validation réside dans la large adoption par un grand nombre d'utilisateurs. Même si l'adoption large d'un outil peut être considérée comme une forme d'évaluation, ce n'est pas forcément une évaluation rationnelle ou scientifique.
Au contraire, certains outils ont fait l'objet d'expériences afin d'évaluer leur contribution. Nous présentons dans cette partie ces expériences. Nous distinguons les études préliminaires des besoins et des pratiques qui motivent la réalisation de la boîte à outils en montrant son utilité et l'importance du problème traité, et des évaluations effectuées pendant et après le développement de l'outil.

1.2.1. Études préliminaires des besoins et des pratiques

Pour plusieurs outils décrits au Chapitre 6, une étude préliminaire des besoins de leurs utilisateurs a été réalisée. Ainsi l'approche adoptée pour ces outils relève de techniques de conception centrée

utilisateurs : Avant de concevoir l'outil, les besoins et les pratiques ont été étudiés. En cela, ils respectent les préconisations présentes dans [Olsen, 2007] et [Greenberg, Buxton, 2007]. Ces utilisateurs sont des « prototypeurs », et leurs expertises sont hétérogènes, du design (arts appliqués) à l'informatique. Nous présentons maintenant les quatre types d'études préliminaires rencontrées.

Tout d'abord, nous trouvons dans [Klemmer et al., 2004] une analyse de 24 interfaces représentatives de celles visées par l'outils. L'objectif était d'identifier des requis pour leur boîte à outils, comme les évènements « ajouter », « mettre à jour » et « supprimer » un objet physique de l'interface. Cette analyse est une des approches préconisées dans [Greenberg, Buxton, 2007].

La seconde forme d'expérience préliminaire est l'entretien. Dans [Klemmer et al., 2004] nous trouvons des entretiens avec des utilisateurs potentiels (développeurs) pour motiver et justifier l'outil. Des entretiens structurés avec neuf concepteurs ayant l'expérience des interfaces tangibles ont été réalisés. Ces concepteurs avaient déjà utilisé la vision par ordinateur, les RFID, des codes barres, etc., c'est-à-dire les modalités d'interaction couvertes par la boîte à outils. Les utilisateurs potentiels ont été interrogés de vive voix, par téléphone, ou par mail. L'objet du questionnaire était les difficultés qu'ils avaient l'habitude de rencontrer.
L'entretien préliminaire est une pratique que l'on retrouve dans [Avrahami, Hudson, 2002] et [Hudson, Mankoff, 2006] : avant de concevoir les boîtes à outils Switcharoo et BOXES, leurs auteurs ont mené des entretiens avec des professionnels du design (arts appliqués).
De même, dans [Hartmann et al., 2006], afin de justifier l'importance et la nécessité de leur contribution, les auteurs reportent la réalisation d'entretiens (individuelles ou en groupe) avec onze professionnels, designers et managers de trois entreprises différentes, ainsi que trois étudiants en design. Les auteurs exposent les commentaires des sujets sur leur activité de prototypage d'objets purement physiques et d'applications purement logicielles, comparées aux interfaces mixtes, ainsi que sur leur façon d'intégrer le prototypage de l'apparence et de l'interaction dans les prototypes.
De même dans [Hartmann et al., 2008], les auteurs ont motivé leur travaux par leur propre expérience de conception, mais aussi d'enseignements. Ils ont également réalisé des interviews de trois designers d'interaction. Ils recueillent les remarques et suggestions des participants, et certaines de leurs pratiques.
L'entretien est une approche préconisée par [Greenberg, Buxton, 2007] (identifications des requis, des aspects culturels et des pratiques existantes, etc.) et [Olsen, 2007] (importance du problème pour des utilisateurs donnés et une situation donnée).

Aux entretiens reportés dans [Avrahami, Hudson, 2002] s'ajoutent l'observation in situ : les auteurs de [Hudson, Mankoff, 2006] et [Avrahami, Hudson, 2002] ont observé des designers pendant leurs enseignements et leurs pratiques. Il s'agit d'une étude ethnographique sur la façon de travailler des designers [Avrahami, Hudson, 2002]. Là encore, cette approche est décrite dans [Greenberg, Buxton, 2007] et [Olsen, 2007].

Enfin, nous trouvons dans [Avrahami, Hudson, 2002] que l'outil a été conçu avec la participation active de quatre designers [Avrahami, Hudson, 2002]. Il s'agit ici de conception participative.

Ces études préliminaires des besoins et des pratiques font partie de la validation d'un outil car elles permettent de souligner leurs apports. Au-delà de ces études préliminaires, certains outils ont été évalués après ou pendant la conception.

1.2.2. Évaluations pendant et après la conception

Nous trouvons plusieurs types d'évaluation d'outils pendant et après leur conception. Le premier type d'évaluation que nous présentons est l'évaluation conceptuelle. Les deux autres types d'évaluation sont expérimentales. Ces évaluations peuvent avoir lieu in situ, dans des situations de conception réelles, ou au contraire en laboratoire.

1.2.2.1. Évaluations conceptuelles

La seule évaluation conceptuelle trouvée parmi les outils présentés au Chapitre 6 se trouve dans [Hartmann et al., 2007]. Les auteurs proposent une évaluation conceptuelle effectuée selon le canevas intitulé « *Cognitive Dimensions of Notations Framework* » [Blackwell, Green, 2000]. Nous avons présenté ce cadre d'évaluation dans la section 2.1 du Chapitre 5 pour l'évaluation du modèle d'interaction mixte. Il se présente sous la forme d'un questionnaire d'inspection de notations [Blackwell, Green, 2000].
Les auteurs de [Hartmann et al., 2007] l'utilisent pour inspecter la notation graphique véhiculée par l'outil. Cette forme d'évaluation est utilisée en complément d'une évaluation expérimentale. En effet, même si elle peut permettre de mettre à jour des problèmes éventuels, cette approche ne peut remplacer l'utilisation concrète d'un outil logiciel dans la situation pour laquelle il a été conçu.

Outre cette forme d'évaluation, les autres évaluations proposées pour les outils étudiés au Chapitre 6 sont expérimentales. Nous présentons d'abord les expériences in situ, dans des situations de conception réelles et sans protocole.

1.2.2.2. Expériences in situ : des situations de conception réelles et sans protocole

Les expériences se déroulant sans protocole dans des situations de conception réelles ne permettent pas d'évaluer l'outil en fonction de variables choisies, mais permettent néanmoins de mettre à jour l'utilisation réelle de l'outil. Leur intérêt réside donc dans la découverte de certains aspects de la boîte à outils, ses atouts mais aussi ses limitations.

Pour Papier-mâché [Klemmer et al., 2004], nous trouvons une évaluation expérimentale de l'*utilisabilité* de la boîte à outils. Cette approche d'évaluation a été réalisée dans un deuxième temps, après une approche suivant un protocole (section suivante, 1.2.2.3). Nous remarquons que, alors que [Greenberg, Buxton, 2007] préconisaient des évaluations moins contrôlées en début de recherche, les auteurs ont fait l'inverse et ont commencé par des évaluations en laboratoire pour finir par des évaluations en situation réelle.
L'évaluation a été réalisée auprès de quatre étudiants. Il s'agissait pour les étudiants de réaliser en une semaine un projet permettant de suivre des pointeurs lasers, capturer des post-its sur un tableau blanc, de lire des codes barres, et d'associer des modalités (vision par ordinateur, RFID, code barre) avec des fonctionnalités de l'application. Ils ont ensuite recueilli les impressions et commentaires des étudiants. Ils ont aussi observé que les étudiants ont pu étendre la boîte à outils fournie.

Pour les Phidgets [Greenberg, Fitchett, 2001], les auteurs ont proposé à seize étudiants, utilisateurs novices de la boîte à outils, l'exercice suivant à effectuer en un temps limité et qui comptait pour 1/10 de leur note finale pour le cours d'Interaction Homme-Machine :
« *A Computer Science professor has designed a variety of phidgets that he plans to demonstrate at a conference. To make this demonstration more interesting, he would like to show how these phidgets could be used in practice. Consequently, he wants you to design an imaginative 'out of the box' interface using these phidgets. The interface you create may be practical, artistic, or fun. It could be geared towards office workers, people at home, children, or whomever you wish.*[50] »
Les auteurs ont alors observé de façon informelle les difficultés des étudiants. Ils ont recueilli les commentaires des étudiants, sur le temps passé à réaliser leur prototype (de quelques heures à quelques jours), ce qui leur avait demandé le plus d'effort (la maquette physique ou la

[50] Un professeur d'informatique a conçu un ensemble de phidgets dont il veut faire la démonstration pendant une conférence. Pour que cette démonstration soit plus intéressante, il voudrait montrer comment ces phidgets peuvent être utilisés en pratique. Par conséquent, il voudrait que vous conceviez une interface innovante en utilisant ces phidgets. L'interface que vous créerez peut être pratique, artistique, ou amusante. Elle peut cibler des employés de bureaux, des gens à la maison, des enfants, ou un autre type de personne de votre choix.

programmation, par exemple). Ils ont aussi observé la façon dont les prototypes conçus ont été accueillis par leurs pairs (par exemple avec des exclamations du type « *wow* »).

De façon similaire pour iStuff [Ballagas et al., 2003], les auteurs présentent des compte-rendu d'utilisation par des utilisateurs novices. Un utilisateur novice a par exemple intégré un dispositif IStuff (le iDog, section 2.2.3.1 page 191) dans l'outil.

Pour BOXES [Hudson, Mankoff, 2006], les auteurs ont mené une évaluation itérative, en deux itérations. La première itération a eu lieu très tôt dans le développement de la boîte à outils, avec un seul designer, dont l'expertise de programmation concernait des environnements comme Flash. Cette expérience a eu lieu sur le terrain. Les auteurs présentent les difficultés rencontrées par ce designer. Par exemple, les auteurs ont dû lui expliquer comment associer deux fonctions au même bouton. La deuxième itération de l'évaluation est une expérience suivant un protocole et est décrite dans la section suivante.

Les auteurs de d.tools [Hartmann et al., 2006] ont évalué une seule partie de leur outil : le mode conception de d.tools, mais pas les deux autres modes présentés (test, analyse). Pour évaluer le mode conception de d.tools, les auteurs ont réalisé trois expériences en sept mois. Les deux premières suivent un protocole, contrairement à la dernière. Comme pour Papier-Mâché, nous remarquons que les auteurs ont commencé par des évaluations en laboratoire pour finir par des évaluations en situation réelle, contrairement à ce qui est préconisé dans [Greenberg, Buxton, 2007].
L'expérience qui nous intéresse ici fut conduite parmi des équipes d'étudiants en cours de conception d'interfaces homme-machine lors de la réalisation de leur projet final. Ils ont eu le choix entre plusieurs outils, dont d.tools, pour réaliser une interface tangible. Les auteurs ont d'abord compté les groupes d'étudiants qui ont choisi d.tools. Pour ceux qui ont choisi d.tools, une assistance technique était fournie. Les auteurs ont observé les groupes et ont recueilli les commentaires. Par exemple, les auteurs ont observé que les étudiants qui n'ont pas choisi d.tools ont écrit beaucoup de code, alors que l'équivalent graphique avec d.tools (qui propose un langage de programmation graphique) aurait été plus concis.

Dans [Truong, 2005] l'évaluation repose sur des cas d'étude d'applications développées par les auteurs (3 applications) ainsi que par d'autres (6 applications). Pour les neuf applications développées avec la boîte à outils, sont décrits :

- Le contexte de l'expérience : projet de recherche, travail d'un étudiant pour le développement d'une application, conception itérative avec des utilisateurs finaux, projet à réaliser en classe comme exercice ou devoir à rendre, projet de Master dont la réalisation s'étale sur plusieurs mois ;
- Les sujets des expériences : les auteurs de l'outil, des étudiants en Master d'informatique ou plus jeunes, souvent travaillant seuls ou en binôme pour la conception et/ou le développement ;
- Leur expérience de la boîte à outils ;
- La version de la boîte à outils utilisée pendant l'expérience ;
- Les éléments de la boîte à outils qui ont été utilisés ;
- Le temps et l'effort nécessaires pour développer l'application ;
- Les remarques et commentaires des utilisateurs ;
- La partie la plus difficile à réaliser pendant le développement de l'application ;
- Les problèmes concernant la boîte à outils qui sont révélés par l'expérience, le cas échéant, et éventuellement les solutions envisagées ou effectivement apportées à la suite de l'expérience.
- Pour l'un des 6 systèmes développés par des étudiants, les auteurs ont ponctuellement conduit des entretiens avec l'étudiant toutes les deux semaines pendant le développement d'une de ces applications. Pendant les entretiens, les auteurs lui demandaient quelles étaient ses avancées, les problèmes qu'il avait rencontrés, ainsi sa méthode de conception. Les auteurs ont également essayé d'identifier le temps qu'il passait sur le projet entre les entretiens. Mais comme l'étudiant reportait le temps pour lequel il était payé, les auteurs

n'ont pas retenu ses réponses comme valides, car elles ne leur ont pas permis pas de savoir quel était le temps effectivement passé sur le développement.

De plus, lors d'une expérience avec seize étudiants utilisant les unes après les autres trois boîtes à outils, les auteurs reportent la moyenne et la variance des notes obtenues par les étudiants pour chaque boîte à outils, ainsi que le nombre d'étudiants n'ayant pas rendu leur devoir. Les auteurs ont également, uniquement pour leur boîte à outils, inspecté la qualité des prototypes rendus par les étudiants.

Nous retenons de ces expériences qu'elles constituent un moyen de mettre un outil de prototypage à l'épreuve pour en identifier les apports et les limites. Ce type d'évaluation permet l'identification de l'importance du problème [Olsen, 2007], l'étude du passage à l'échelle de l'outil [Olsen, 2007], ainsi que l'évaluation in situ préconisée par [Greenberg, Buxton, 2007]. Même s'il n'est pas forcément possible d'interpréter correctement ou de généraliser les résultats d'expériences de ce type, nous souhaitons mettre en place cette forme d'évaluation car elle nous paraît constructive.

Nous présentons maintenant les expériences effectuées en laboratoire, avec un protocole et en présence d'expérimentateurs.

1.2.2.3. Expériences en laboratoire avec un protocole

Nous considérons d'abord les études qui consistent à re-créer des applications existantes pour démontrer la puissance d'un outil (« plafond » de [Myers et al., 2000]). Dans ces expériences, ce sont les auteurs eux-mêmes qui utilisent l'outil et qui choisissent les applications à développer. Par exemple, les auteurs de la Context Toolkit [Salber et al., 1999] présentent des applications développées, en précisant les lignes de code utilisant la boîte à outils par rapport au nombre de lignes de code total écrites pour développer l'application. Les applications développées sont soit nouvelles, soit issues de la littérature, soit une application existante modifiée. Ainsi les auteurs veulent prouver que la boîte à outils permet le développement des applications « classiques » et peut être utilisée pour une application déjà existante, mais aussi qu'elle permet le développement de nouvelles applications. C'est une évaluation de la puissance de l'outil [Myers et al., 2000].

La seconde expérience de ce type se trouve dans [Hartmann et al., 2006]. Les auteurs ont re-prototypé des applications existantes et prototypé de nouvelles interfaces. Ils ont ainsi observé comment d.tools passe à l'échelle de systèmes plus complexes. Ils ont également mesuré le temps passé aux différentes activités.

De même dans [Ballagas et al., 2007], les auteurs montrent la puissance de leur outil en présentant plusieurs exemples de prototypes de complexités différentes. Ils montrent par un ensemble d'exemples la capacité de l'outil à recréer au moins les interactions existantes, mais aussi sa capacité à les recréer plus facilement.

Nous considérons ensuite les études expérimentales avec des utilisateurs novices. Ces expériences peuvent nécessiter un nombre important de sujets utilisateurs et un protocole défini. Elles sont donc coûteuses en temps, mais permettent d'évaluer l'outil en fonction de variables précises, et souvent permettent d'estimer comment les apports de la boîte à outils sont ou peuvent être atteints.

Une partie de l'évaluation de Papier-Mâché [Klemmer et al., 2004] s'est déroulée de cette façon, en laboratoire et sous contrôle des auteurs. Le but de l'expérience était d'évaluer l'*utilité* des choix faits pour la boîte à outils. Les sujets étaient sept étudiants en informatique, mais non spécialistes en Interaction Homme-Machine. Ils étaient tous expérimentés en java, langage de base de la boîte à outils.

Pendant l'expérience, chaque session consistait en une démonstration de l'outil et de la fenêtre de contrôle de Papier-Mâché (Chapitre 6, section 2.2.2). Ils demandaient ensuite aux sujets de lire un manuel utilisateur. Ensuite, ils demandaient aux sujets de réaliser une première tâche facile pour commencer : modifier une application déjà développée, pour qu'elle ne cherche plus les objets rouges mais les objets bleus. Enfin, deux tâches plus difficiles étaient proposées. La première tâche consistait à modifier une application déjà développée pour qu'elle utilise la technologie RFID à la place de la vision par ordinateur. La seconde tâche encore plus difficile consistait à développer une

application qui utilisait les marqueurs RFID pour contrôler un diaporama : un marqueur devait représenter le dossier contenant les photos dans l'arborescence de fichiers de l'ordinateur, et les deux autres marqueurs devaient représenter les fonctions « précédente » et « suivante ». Pendant l'expérience, les auteurs n'ont pas répondu aux questions des sujets sur l'outil évalué, mais renvoyaient les sujets vers la documentation. Il a aussi été demandé aux participants de penser à haute voix. Les sessions ont été enregistrées par vidéo.
Le code produit a été inspecté après chaque session. Les auteurs ont aussi relevé les commentaires des participants. Ils ont également mesuré le temps mis pour réaliser les différentes tâches, ainsi que le nombre de lignes de code produites, et ont observé l'évolution de ces chiffres afin de mesurer l'apprentissage. Il est intéressant de souligner ici que les résultats de cette étude ont montré que la première tâche a finalement été la plus difficile pour les sujets, à cause de la nécessité de changer la couleur codée dans l'espace IHS, beaucoup moins habituel pour les programmeurs que l'espace RGB. Cela a permis aux auteurs d'observer un peu plus l'utilité de la fenêtre de contrôle fournie par leur boîte à outils.

Comme annoncé à la section précédente, les auteurs de d.tools [Hartmann et al., 2006] ont réalisé trois expériences, la dernière étant une expérience sans protocole et reportée précédemment. Les deux premières expériences, que nous présentons maintenant, ont eu lieu en laboratoire. Pour la première expérience, treize designers et étudiants en design ont participé. Les auteurs leur ont demandé de réaliser trois prototypes de portée croissante en une heure et demie. Ils leur ont ensuite demandé de répondre à un questionnaire. Le questionnaire leur demandait de noter la capacité de d.tools à « permettre les tests d'utilisabilité, raccourcir le temps requis pour construire un prototype, et aider la compréhension de l'expérience de l'utilisateur » pendant la phase de conception. Nous notons que ce questionnaire proposait des questions déjà orientées, par exemple avec la formulation « raccourcir le temps » au lieu de demander si « le temps requis pour construire un prototype est (1) plus long ou (2) moins long ».
Les auteurs reportent également leurs observations : les difficultés rencontrées, ce que les utilisateurs ont apprécié, mais aussi si le travail s'est principalement porté sur l'apprentissage de l'outil évalué ou sur la tâche. Cette première expérience avant pour but d'évaluer la difficulté de prise en main [Myers et al., 2000].

Pour BOXES [Hudson, Mankoff, 2006], après amélioration de la boîte à outils grâce aux retours de la première évaluation en situation réelle (décrites à la section précédente), les auteurs ont effectué une deuxième expérience. Cette expérience en laboratoire impliquait trois designers. Deux designers savaient programmer en Flash, le troisième ne programmait pas et n'utilisait que Photoshop ou le papier pour prototyper. La première étape de l'expérience consistait en une démonstration de la boîte à outils pour la conception d'un système simple. Ensuite il a été demandé aux trois sujets de prototyper un autre système simple avec la boîte à outils et des matériaux comme du carton, des punaises, etc. Les auteurs ont relevé des informations sur le nombre d'itérations effectuées par les designers, pour la partie physique des prototypes et pour l'interaction avec les prototypes. Ils ont noté ont testé leurs prototypes, mais aussi quand les designers ont effectué des actions du type : couper le carton, attacher les punaises, construire l'interaction en connectant les punaises à des fonctionnalités, etc. Ils ont aussi noté les difficultés des designers pour utiliser l'interface proposée et l'aide qu'ils ont dû apporter au designer qui ne savait pas programmer. Ils ont observé vers quels sujets se portait l'attention des designers : l'outil évalué (interface matérielle et logicielle à l'écran) ou la conception du prototype. Les auteurs reportent enfin les remarques des designers (1) à propos de l'utilité de l'outil, et (2) à propos des limitations de l'outil.

Dans [Hartmann et al., 2007], nous trouvons une étude en laboratoire des premières utilisations de Exemplar. Douze sujets ont participé à l'expérience. Les participants avaient des niveaux de qualifications et des expertises différentes (étudiants à différents niveaux, en Informatique/Interaction Homme-Machine, ingénierie, éducation ou sciences humaines). Tous avaient déjà l'expérience de la programmation avec des capteurs, mais aucun n'était expert. Les auteurs leur ont demandé leur expertise en capteurs, en design physique et en programmation.

Les sujets étaient assis en face d'un ordinateur à deux écrans et équipé de l'interface matérielle d'Exemplar. Un des écrans affichait l'interface logicielle d'Exemplar, l'autre une documentation sur les capteurs.
Les sessions duraient deux heures. Dans un premier temps, les expérimentateurs faisaient une démonstration d'Exemplar. Dans un second temps leur étaient présentés les capteurs. Ensuite, les participants ont réalisé trois exercices. Pour chaque exercice, les évènements produisant les sorties demandées étaient fournis, afin que les sujets se concentrent sur l'interaction en entrée avec les capteurs. La première tâche consistait à afficher « *Hello* » quand un capteur de pression était pressé au-dessus d'un certain seuil, et à afficher « *World* » quand un autre capteur de pression était pressé trois fois de suite. La seconde tâche consistait à faire clignoter à gauche ou à droite une simulation de casque de vélo lorsqu'un vrai casque de vélo était penché vers la gauche ou la droite. La troisième tâche était plus ouverte : il s'agissait de concevoir un dispositif de contrôle pour un de deux jeux vidéos proposés. Pour l'un des jeux, il fallait contrôler la position d'un vaisseau, pour l'autre, il fallait viser et tirer sur des cibles.
Après avoir fini, les participants ont répondu à un questionnaire. Sur une échelle de cinq valeurs, ils devaient dire si Exemplar :

- diminuait le temps nécessaire pour construire des prototypes,
- les faisait expérimenter davantage,
- facilitait les modifications rapides,
- leur enseignait comment un capteur fonctionne,
- les aidait à comprendre l'expérience de l'utilisateur,
- les aidait à mener des tests d'utilisabilité,
- incitait des solutions de conception plus innovantes et adéquates,
- diminuait le temps nécessaire pour tester les prototypes,
- les éloignait de leur tâche de conception,
- leur faisait construire moins de prototypes,
- rallongeait le temps nécessaire pour programmer l'interaction avec les capteurs.

Là encore, nous soulignons que ce questionnaire proposaient des questions déjà orientées. De plus, cette forme d'évaluation avec beaucoup plus de sujets que la précédente ne permet pas de faire débattre les sujets sur l'outil.

Nous trouvons aussi une évaluation de ce type dans [Ballagas et al., 2007]. Pour montrer que le seuil de prise en main de leur outil [Myers et al., 2000] est bas, ils utilisent des métriques comme le temps de développement et le nombre d'itérations, sans considérer le nombre de lignes de code puisque le prototypage se fait graphiquement. Ici, contrairement aux précédentes évaluations, les auteurs ont choisi un outil de référence (le Patch panel de iStuff [Ballagas et al., 2003]) afin de comparer les deux outils sur une tâche similaire. En effet, le Patch panel (outil de référence) est un équivalent textuel à l'outil graphique (outil évalué). Afin de réduire le biais dû à la tâche demandée, les auteurs ont contrebalancé les tâches en faisant réaliser quatre tâches différentes. Seize sujets ont participé à l'expérience. Les sujets étaient étudiants, en moyenne en quatrième année d'études universitaires et avaient suivi des cours d'Interaction Homme-Machine. Le guide d'animation de l'expérience prévoyait une alternance de formation et d'exercices durant 3h30 (Tableau 26). L'activité des participants est arrêtée au bout de 30 minutes, qu'ils aient terminé leur prototype ou non.

Début	*Fin*	**Activité**
0h00	*0h30*	Introduction à iStuff
0h30	*1h00*	Formation au 1er outil
1h00	*1h30*	Exercice 1
1h30	*2h00*	Exercice 2
2h00	*2h30*	Formation au 2ème outil
2h30	*3h00*	Exercice 3
3h00	*3h30*	Exercice 4

Tableau 26: Guide d'animation de l'expérience pour valider iStuff mobile (d'après [Ballagas et al., 2007]).

Les sujets ont été repartis dans deux groupes A et B de huit personnes. Le Tableau 27 montre comment les auteurs ont contrebalancé les outils (outil de référence et outil évalué) selon les groupes A et B de sujets. Dans chaque groupe, les sujets ont travaillé par équipe de deux. Chaque équipe a réalisé les exercices dans un ordre différent (technique du carré latin).

	1er outil	**2ème outil**
Groupe A	Outil évalué	Outil de référence
Groupe B	Outil de référence	Outil évalué

Tableau 27: Contrebalancer les outils selon les groupes de sujets pour l'évaluation de iStuff mobile.

Les auteurs ont mesuré le temps nécessaire à la construction du premier prototype (c'est-à-dire l'implémentation d'au moins une partie des fonctionnalités demandées par la tâche), ainsi que le nombre d'itérations effectuées (c'est-à-dire le nombre de cycle conception/test/analyse).
Les auteurs présentent ensuite le temps moyen nécessaire à la construction du premier prototype pour chaque outil en testant la significativité. Ils présentent aussi la proportion du nombre d'itérations ≥ 1 pour chaque outil, ainsi que le nombre d'itérations moyen pour chaque outil, et concluent que leur outil (outil graphique évalué) est significativement plus rapide que l'autre (Patch panel, outil de référence), même si la comparaison est limitée (16 sujets, 4 tâches, 30 minutes, etc.).

Dans [Hartmann et al., 2008b], nous trouvons une évaluation de l'outil Juxtapose avec 18 participants. Les participants étaient des étudiants ayant des connaissances en Interaction Homme-Machine et de l'expérience en conception d'interaction. Tous les participants, sauf un, avaient l'expérience de la programmation. Les sessions d'évaluation duraient 75 minutes. Les participants étaient installés à une station de travail avec une souris un clavier, et un dispositif avec des glissières contrôlables par l'utilisateur et motorisées (contrôlables par la machine). Les expérimentateurs ont d'abord fait une démonstration de l'outil. Les participants ont ensuite réalisé 3 tâches :

1. Pour la première tâche, les participants devaient faire des modifications nécessitant d'utiliser les deux fonctionnalités présentées comme contribution de l'outil Juxtapose : la production simultanée de solutions de conception et l'ajustement des variables pour la mise au point des solutions de conception.
2. Pour la deuxième tâche, les participants devaient trouver les paramètres adéquats pour prototyper quatre solutions de conception demandées. L'ordre des exercices a été tiré a sort pour chaque participant. De plus, pour comparer l'outil à un outil de référence, deux solutions de conception ont été réalisées avec Juxtapose (outil évalué), tandis que les deux autres solutions de conception ont été réalisées avec Juxtapose privé des fonctionnalités présentées comme contribution par les auteurs (outil de référence). L'outil utilisé (outil évalué ou outil de référence) a été contrebalancé pour les quatre exercices.
3. Pour la troisième tâche, les participants devaient créer en 30 minutes deux alternatives de conception à partir d'un programme de navigation dans une carte qui possédait 28 types d'information (rues, autoroute, parcs, etc.). Ils avaient à leur disposition une documentation pour trouver comment faire. Ils ont ensuite présenté leur travail et leurs résultats à l'expérimentateur.

Les auteurs ont ensuite recueilli les commentaires des participants (résultats qualitatifs). Ils ont aussi fait une étude quantitative afin de voir si les participants ont pu explorer plus de paramètres de conception, et s'ils les ont explorés plus rapidement. Pour la seconde tâche, ils reportent le temps moyen de complétion des exercices selon qu'ils disposaient de l'outil évalué ou de l'outil de référence et ont effectué un test de significativité. Ils reportent aussi ce temps en fonction de l'exercice et expliquent les différences. Pour quantifier le coût d'une modification, les auteurs ont compté le nombre de combinaisons de paramètres explorées (c'est-à-dire, dans le cas de l'outil de référence, le nombre d'exécution du code après le changement du code source ; et dans le cas de l'outil évalué, le nombre d'évènements d'ajustements envoyés depuis l'interface d'ajustement). Les auteurs reportent des limitations pour l'utilisabilité, venant de leur observation ou de suggestions des participants.

Nous venons de présenter les évaluations en laboratoire effectuées par les auteurs des outils logiciels présentés au Chapitre 6. Même si elles partagent le fait de suivre un protocole, ces expériences ne sont pas toutes de même nature : certaines sont quantitatives, d'autres qualitatives ; certaines avaient un protocole très contrôlé, d'autres moins ; certaines ont été réalisées avec un grand nombre de sujets (18 pour la dernière), d'autres avec peu (3 pour BOXES). Nous présentons maintenant notre analyse de ces évaluations.

1.2.3. Analyse des méthodes d'évaluation

Ces formes d'évaluation d'outils soulignent la difficulté de l'entreprise car la conception et le prototypage sont des tâches complexes et difficiles à mesurer de façon non-ambiguë.

Les études préliminaires constituent la première forme d'évaluation que nous avons présentée. Elles sont intéressantes pour démontrer l'utilité et l'importance du problème traité par l'outil. En cela, elles respectent les préconisations de [Olsen, 2007] et celles de [Greenberg, Buxton, 2007].
Les études in situ et sans protocole sont essentielles et permettent d'évaluer l'utilisation de l'outil en situation réelle. Pourtant, elles ne permettent pas forcément à l'expérimentateur d'évaluer un aspect particulier de l'outil. Nous pensons qu'elles sont un bon moyen d'évaluer la puissance de l'outil (« plafond » de [Myers et al., 2000]) et le passage à l'échelle qui est une lacune éventuelle identifiée par [Olsen, 2007].
Au contraire, les études en laboratoire et suivant un protocole sont contraignantes : elles demandent souvent beaucoup de moyens (recrutement de sujets, définition d'un protocole, etc.) et manquent de réalisme. Pourtant, elles sont utiles pour évaluer des facteurs précis. Dans ce cadre, les expérimentateurs peuvent choisir de mener une évaluation quantitative ou qualitative. Dans le premier cas, les expérimentateurs cherchent à mesurer des variables comme le nombre de lignes de code produites par les sujets. Pourtant, ces études quantitatives sont difficiles à interpréter : par exemple, le nombre de lignes de code ou le temps de développement est dépendant de l'expertise des sujets, mais aussi de leur habitude et de leur préférence. Les études qualitatives ne permettent pas de mesurer des variables, mais permettent tout de même de mettre à jour les points positifs et les limitations d'un outil. C'est par exemple l'évaluation de BOXES [Hudson, Mankoff, 2006]. Cette forme d'évaluation permet de mettre des utilisateurs novices dans une situation de réalisation d'un exercice proposé. Ce type d'expérimentation permet aussi de recueillir de nombreux retours sur l'outil. Il nous semble que c'est une forme d'évaluation adaptée à notre outil, car nous cherchons à évaluer OP auprès de novices, tout en ne maîtrisant pas encore ses limites. Clairement, les deux formes d'évaluation, avec ou sans protocole, sont complémentaires.

Nous soulignons également que l'ordre de réalisation des évaluations avec ou sans protocole varie. Par exemple pour BOXES [Hudson, Mankoff, 2006], les auteurs ont d'abord effectuées une évaluation in situ puis une évaluation en laboratoire, alors que pour évaluer Papier-Mâché [Klemmer et al., 2004], l'évaluation en laboratoire a eu lien avant l'évaluation in situ. Conformément au cadre d'évaluation de [Greenberg, Buxton, 2007] nous préférons évaluer d'abord de façon sans protocole avec d'identifier les points positifs et limitations de l'outil, puis passer à une évaluation en laboratoire pour mieux cerner ses apports et ses limites.

Enfin, il est rare de trouver une étude comparative de deux outils. La difficulté d'une évaluation comparative est soulignée dans [Olsen, 2007] et [Greenberg, Buxton, 2007]. De plus certains outils ne sont pas disponibles (Chapitre 6, section 2.3).

Dans le cadre de cette thèse, nous considérons une évaluation qualitative de l'outil OP. Les expérimentations de type quantitatives font partie des perspectives à nos travaux.

2. Carnet de route et évaluations de la boîte à outils OP

Vis-à-vis des approches d'évaluation d'outils pour l'Interaction Homme-Machine que nous avons présentées, nous avons choisi de présenter une évaluation conceptuelle de notre outil OP, puis des études expérimentales de son utilisation, dans des situations réelles puis en laboratoire.

2.1. Évaluation conceptuelle de la boîte à outils OP

Pour valider OP, nous expliquons d'abord l'importance du problème que OP cherche à résoudre [Olsen, 2007], puis nous exposons notre démarche de conception, selon la validation préconisée par [Greenberg, Buxton, 2007] et présentée à la section 1.1.4.

2.1.1. Importance du problème traité par la boîte à outils OP

Selon [Olsen, 2007], il convient de justifier l'importance du problème traité par la boîte à outils OP, pour les utilisateurs cibles et la situation ciblée.
Le problème traité par la boîte à outils OP est l'opérationnalisation du modèle d'interaction mixte afin qu'il puisse être utilisé pendant les phases conceptuelles et pratiques de la conception.
La population visée est celle des concepteurs d'interfaces (designers, ergonomes, informaticiens). Pour le moment, OP s'adresse à des utilisateurs qui savent écrire des lignes de code, mais nous souhaitons étendre OP pour éviter d'écrire des lignes de code. Nous justifions donc de l'importance du problème pour les concepteurs quelle que soit leur expertise.
La situation traitée par la boîte à outils OP est la conception d'interfaces mixtes par un concepteur ou un groupe de concepteurs.

Pour les informaticiens, la tâche de prototypage ou réalisation d'ébauches est importante. Nous l'avons constaté nous-même dans notre pratique, mais aussi dans la littérature (Chapitre 6). Pour justifier l'importance du prototypage lié à la conception, nous avons également considéré les pratiques existantes auprès d'autres concepteurs d'autres domaines que le nôtre (non-informaticiens) et nous avons constaté qu'ils sont exposés aux mêmes problèmes. Nous avons discuté et observé des artistes des nouveaux médias et des designers d'interaction sur leur pratique de conception. Nous avons remarqué qu'ils n'ont pas non plus de pratique consensuelle et qu'ils réalisent des ébauches interactives de façon ad-hoc car ils en ont besoin. Un designer d'interaction nous raconte : « J'ai eu ce genre de discussions tout au long de l'année dernière sur le grand projet de diplôme avec [mon collègue], on voulait à la fois pouvoir concevoir le projet et l'expliquer aux profs […]. On a testé des trucs sur eux pour expliquer nos projets: alors dans ce qui a marché: les romans-photos, les films, les prototypes rapides, les jeux et dans ce qui ne fonctionne pas: les schémas entrées/sorties "cybernétique", l'aspect théorique et l'aspect technique (quels programmes, quels composants, etc.). J'ai fait l'amère expérience de présenter le projet devant des personnes qui ne le connaissent pas du tout, et en montrant les schémas "abstraits" des interactions, et tout le monde s'endort !! Je pense les trucs trop abstraits ne permettent pas de bien décrire certains systèmes, plus exactement ils les décrivent tellement bien qu'ils en perdent tout leur côté "humain". […]
Paradoxalement ce genre de représentations abstraites […] nous ont aidées à concevoir le projet […]. On a passé beaucoup de temps à faire des classifications et des abstractions pour finalement faire des visualisations (trucages photos, films, graphiques, etc.) et des simulations (prototypes maquettes) de nos idées puis de nos objets. Les objets que l'on a présentés au diplôme, on les voyait plus comme des expériences que comme des objets au sens du design industriel, et pas comme des

installations artistiques non plus. On a compris aussi au fur et à mesure de l'année, vu qu'on ne connaissait pas grand-chose au design d'interaction, que pas mal de gens procèdent comme ça dans les labos, les boîtes »
Nous soulignons donc que ce designer a eu besoin d'un modèle conceptuel, mais aussi de prototyper, et que passer de l'un à l'autre leur a pris de temps (« On a passé beaucoup de temps à faire des classifications et des abstractions pour finalement faire [...] prototypes maquettes »). Lorsque je lui explique ce que je fais, il me répond que « c'est exactement ça "l'exploration des possibilités, conceptuellement mais aussi par l'expérimentation et le prototypage, au plus tôt et le plus rapidement pour pouvoir rebondir dessus", pour provoquer les discussions pertinentes, du point de vue éthique, de la faisabilité, la cohérence ou l'incohérence... c'est tout à fait ça ! »
Cette discussion souligne que ce problème du prototypage lié à la conception n'est pas important uniquement pour les concepteurs informaticiens, mais pour les autres domaines aussi. Les prototypes ou ébauches servent à communiquer les choix de conception, mais aussi à enrichir la conception en servant de support matériel aux alternatives de conception. De plus, nous avons souligné au Chapitre 6 les limitations des outils existants au regard de notre objectif d'opérationnalisation du modèle d'interaction mixte.

Nous avons identifié l'importance du problème traité par notre outil OP. Nous présentons maintenant la démarche de conception de la boîte à outils OP (validation conceptuelle préconisée par [Greenberg, Buxton, 2007]).

2.1.2. Démarche de conception de la boîte à outils OP

Comme expliqué dans [Greenberg, Buxton, 2007] (section 1.1.4), nous expliquons notre démarche de conception. Énoncer notre démarche de conception permet de rendre explicite les questions centrales de conception de l'outil OP comme celles de tout outil de prototypage, les alternatives que nous avons envisagées pour répondre à ces questions, ainsi que les critères que nous avons utilisés pour raisonner.

Nous rappelons tout d'abord que la démarche de conception que nous avons adopté pour concevoir notre outil de prototypage prend en compte des aspects conceptuels et pratiques. Nous représentons cette démarche à la Figure 221.

- Nous avons intégré les concepts du modèle d'interaction mixte.
- Nous avons pris le parti de :
- capitaliser des caractéristiques pertinentes remarquées dans les outils de prototypage existants, comme l'interface graphique pour la simulation (Magicien d'Oz) qui est présente dans d.tools ou les Phidgets,
- d'intégrer complètement des outils existants (Interface-Z, ARToolKit et Phidgets).
- Nous avons basé notre travail sur notre propre expérience de prototypage des interfaces mixtes. Partant du constat que les prototypes étaient difficilement réutilisables, nous avons donc voulu capitaliser les éléments de prototypage le plus souvent rencontrés.

Nous avons ainsi construit une première version de la boîte à outils, puis au fil des expériences de prototypage, nous l'avons améliorée de façon itérative (boucle utilisation-expérience de la Figure 221).

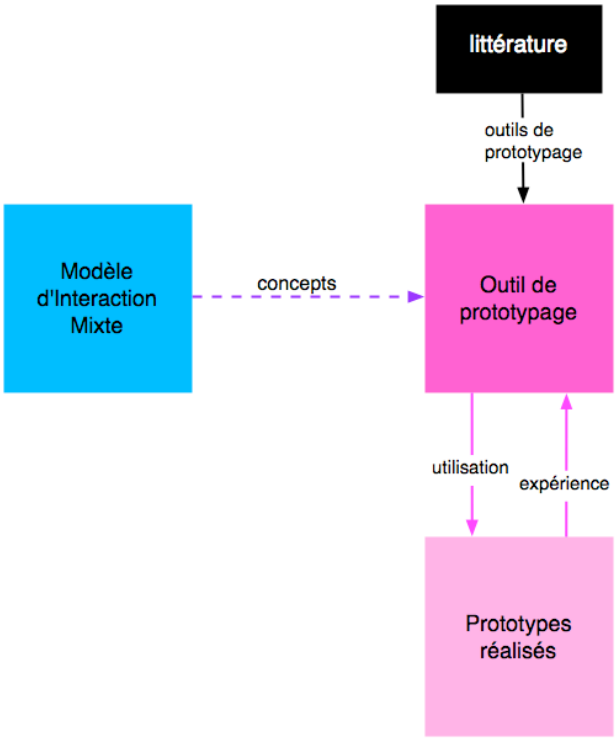


Figure 221 : Démarche de conception de l'outil OP.

Bien que nous n'ayons pas produit d'alternatives de conception pour l'outil OP, nous avons néanmoins considéré des alternatives à chaque étape.
Tout d'abord, nous avons étudié les outils existants et montré qu'ils n'intégraient pas les concepts du modèle d'interaction mixte (Chapitre 6). Ceci justifie la conception de l'outil OP.
Pour concevoir OP, nous avons envisagé d'autres solutions, comme étendre un outil existant (par exemple d.tools). Nous avons préféré prendre le parti d'étendre une boîte à outils graphique afin d'amener plus facilement des utilisateurs de l'outil graphique aux interfaces mixtes. Les contributions des outils existants (par exemple, pour d.tools, la prise en compte de la phase d'évaluation du prototype dans l'outil de conception) pourront être intégrées comme des perspectives à ces travaux.
Nous aurions pu également prendre le parti d'étendre une autre boîte à outils graphique que Qt, comme Java Swing. Comme le montre notre enquête auprès des personnels du Laboratoire d'Informatique de Grenoble (section 2.3.1.2), les deux outils sont utilisés parmi les développeurs d'interfaces graphiques. L'utilisation de ces outils ne nous permet donc pas de choisir l'un ou l'autre. Développer les deux alternatives serait trop coûteux en temps dans le cadre d'une thèse, et nous ne sommes pas sûrs que la différence serait significative.
Parmi les nombreux outils que nous aurions pu intégrer dans OP, comme Arduino et ARToolKit (Chapitre 6), nous avons choisi des outils populaires auprès de différentes communautés d'utilisateurs (artistes, Interaction Homme-Machine, réalité augmentée) en évitant les redondances : Interface-Z, Phidgets et ARToolKit. Il serait intéressant de considérer le développement d'autres alternatives (perspectives à ces travaux). Nous sommes par exemple en cours d'intégration d'Arduino dans OP, car cet outil est également très utilisé dans la communauté des artistes.

Enfin, nous rappelons que les évaluations de l'outil qui ont été conduites sont basées sur une étude des approches d'évaluation d'outils pour l'Interaction Homme-Machine.

Après cette justification conceptuelle selon le modèle de [Greenberg, Buxton, 2007], nous exposons les études expérimentales en situation réelle que nous avons menées. Celles-ci constituent le carnet de route de l'élaboration de OP, car elles nous ont permis d'améliorer l'outil au fil des itérations.

2.2. Carnet de route et études expérimentales en situation réelle d'utilisation de la boîte à outils OP

Conformément à la préconisation de [Olsen, 2007] et [Greenberg, Buxton, 2007] (section 1.1.4), nous avons d'abord conduit des études expérimentales de l'utilisation de OP en situations réelles de conception. Ces premières évaluations ont eu pour but de mettre à l'épreuve les premières versions de la boîte à outils. Ces expériences ont consisté à prototyper et concevoir des interfaces mixtes avec des designers. Nous avons d'abord fait l'expérience avec ORBIS, puis intégré les résultats de cette première expérience avant de faire l'expérience avec Roam. Nous présentons maintenant ces deux expérimentations.

2.2.1. Étude expérimentale pour le système ORBIS

ORBIS est un système que nous avons prototypé et conçu en collaboration avec un designer. Nous avons présenté l'utilisation du modèle d'interaction mixte sur l'exemple d'ORBIS au Chapitre 5 (section 3.1.2.1). Un designer en binôme avec un expert (auteur de la boîte à outils) ont participé à ces expériences. Nous décrivons ici les aspects liés au prototypage lors des séances de conception. L'objectif était plus ici d'evaluer la puissance (« plafond » [Myers et al., 2000]) que la difficulté de prise en main de l'outil (« seuil » [Myers et al., 2000]). Nous présentons ensuite nos conclusions sur cette expérience.

2.2.1.1. Description

Nous avons présenté au Chapitre 5 l'exploration de l'espace de conception avec le modèle d'interaction mixte. Cette exploration a été complétée par des activités de prototypage (ou réalisation d'ébauches) que nous présentons ici. Pendant la conception d'ORBIS, nous avons pour la première fois utilisé OP pour prototyper pendant la conception.
Nous rappelons que ORBIS est un système pour visionner des photos. La liste de photos est importée dans le système au préalable. Les photos sont embarquées dans le système et affichées sur un petit écran. Quelle que soit l'orientation de l'objet dans le plan vertical, les photos sont toujours redressées par le système grâce à des accéléromètres qui captent l'orientation de l'objet. L'objet de la tâche de l'utilisateur est la liste de photos, que nous avons prototypé. La Figure 222 présente le prototype physique de la liste de photos et la Figure 223 présente le logiciel développé avec OP. Ce prototype utilise les composants OP *MIDIDevice* (lignes 4 et 7), *Complementarity* (ligne 11), *OrientationInputLanguage* (ligne 14) et *SlideshowOutputLanguage* (ligne 18) pour les modalités de liaison. Nous soulignons que dans cette première version d'OP, le prototype utilise un composant OP sur mesure pour les propriétés numériques (ligne 16). Ce composant sur mesure a été réalisé avant l'utilisation expérimentale de la boîte à outils OP.

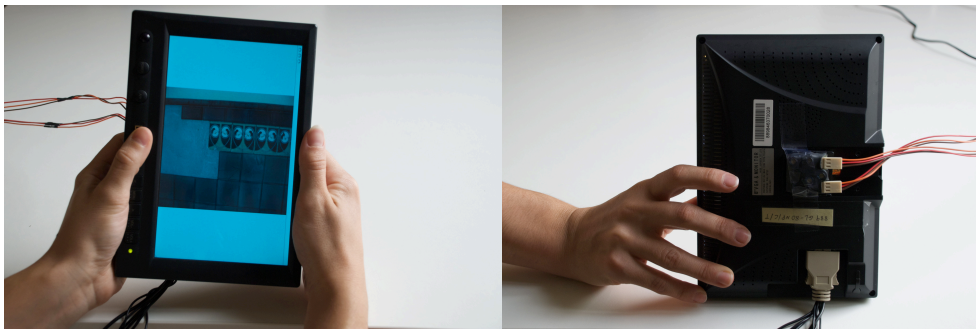


Figure 222: Prototype matériel de l'objet de la tâche, développé avec OP.

```
1  int indexAccelerometer1 = 2;
2  bool isAnalogue = true;
3  int deviceResolution = 7;
4  MIDIDevice accelerometer1("accelerometer1", LinkingComponent::IN, indexAccelerometer1, isAnalogue, deviceResolution);
5
6  int indexAccelerometer2 = 3;
7  MIDIDevice accelerometer2("accelerometer2", LinkingComponent::IN, indexAccelerometer2, isAnalogue, deviceResolution);
8
9  int nbOfConnections = 2;
10 int time = 1000;
11 Complementarity shakeComplementarity("shakeComplementarity", nbOfConnections, time, Composition::DEVICE);
12
13 int orientationResolution = 4;
14 OrientationInputLanguage orientationIL("Orientation", orientationResolution);
15
16 PhotoListDigitalProperties myPhotoListDP("MyPhotoList");
17
18 SlideshowOutputLanguage slideshowOL("slideshow");
19
20 QObject::connect(&accelerometer1, SIGNAL(updated(int, QTime, QString)),
21                  &shakeComplementarity, SLOT(update1(int, QTime)));
22 QObject::connect(&accelerometer2, SIGNAL(updated(int, QTime, QString)),
23                  &shakeComplementarity, SLOT(update2(int, QTime)));
24 QObject::connect(&shakeComplementarity, SIGNAL(updated(QVector<int>, QTime)),
25                  &orientationIL, SLOT(update(QVector<int>, QTime)));
26 QObject::connect(&orientationIL, SIGNAL(updated(int, QTime)),
27                  &myPhotoListDP, SLOT(updateTop(int)));
28 QObject::connect(&myPhotoListDP, SIGNAL(CurrentPhotoUpdated(QImage*, int)),
29                  &slideshowOL, SLOT(updatePhoto(QImage*, int)));
```

Figure 223: Code OP correspondant au prototype de la Figure 222.

ORBIS permet à l'utilisateur d'interagir avec cet objet : il permet notamment de démarrer ou arrêter la présentation des photos, naviguer dans les photos et mélanger les photos, grâce à des outils dont nous avons réalisé des ébauches. Nous en détaillons la réalisation.

2.2.1.1.1. Exploration pratique des possibilités pour l'outil pour naviguer

Nous présentons la conception pratique de l'outil pour naviguer dans la liste de photos. La Figure 224 présente le prototype physique final de l'outil pour naviguer dans la liste de photos. La Figure 225 présente le code OP correspondant. À la ligne 4, nous utilisons un composant *MIDIDevice* pour un potentiomètre d'Interface-Z. Nous le connectons (lignes 16 et 17) à un composant pour le langage de liaison *IdentityInputLanguage* (ligne 9). Cette modalité de liaison est connectée (lignes 18 et 19) à des propriétés numériques prototypées par le composant *TurnableObjectDigitalProperties* (ligne 11). Ce composant de la première version de la boîte à outils a dû être réalisé sur mesure pour le prototype. Il contient les propriétés *angle* et *level* qui sont des entiers. À ces propriétés numériques est connectée (lignes 20 et 21) un composant *BeepOutputLanguage* (ligne 14) pour la modalité de liaison en sortie.

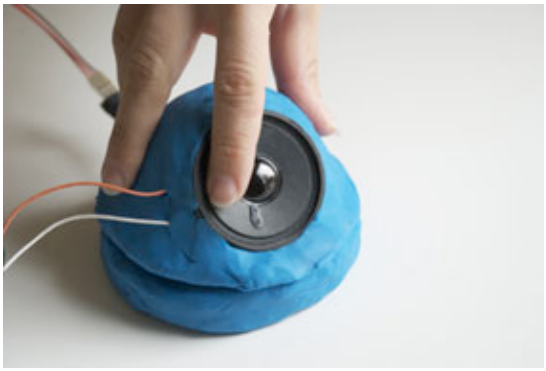


Figure 224: Prototype physique final de l'outil pour naviguer dans la liste de photos.

```
int indexOnBoard = 1;
bool isAnalogue = true;
int resolution = 7;
MIDIDevice TurnSensor("TurnSensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);

bool isOpposite = false;
int hardMin = 0;
int hardMax = 127;
IdentityInputLanguage TurnIL("Turn Input Language", isOpposite, hardMin, hardMax);

TurnableObjectDigitalProperties MyTurnableObjectDP("MyTurnableObject");

char* soundFile = "./Tink.aiff";
BeepOutputLanguage TurnBeepOL("TurnBeepOL", soundFile);

QObject::connect(&TurnSensor, SIGNAL(updated(int, QTime)),
                 &TurnIL, SLOT(update(int, QTime)));
QObject::connect(&TurnIL, SIGNAL(updated(int, QTime)),
                 &MyTurnableObjectDP, SLOT(updateAngle(int)));
QObject::connect(&MyTurnableObjectDP, SIGNAL(LevelUpdated(int)),
                 &TurnBeepOL, SLOT(update()));
```

Figure 225: Code OP l'outil pour naviguer dans la liste de photos.

Avant de définir ce prototype final, nous avons exploré l'espace de conception en prototypant des ébauches interactives. Nous avons pendant la conception d'ORBIS considéré à la fois l'apparence et l'interaction, mais nous ne discutons ici que de l'interaction.
Nous avons trouvé par l'expérience que les propriétés physiques de la Figure 224 incitaient à presser ou à tourner. Nous avons donc décidé de capturer la pression ou la rotation.
Nous avons construit un prototype de l'outil avec un dispositif de liaison qui capte la rotation et un autre avec un dispositif qui capte la pression. La Figure 226 présente le prototype avec un potentiomètre MIDI d'Interface-Z captant la rotation. Figure 227 présente le prototype avec un capteur de pression atmosphérique Interface-Z. Comme ces deux dispositifs peuvent être connectés à la même entrée de la carte physique vendue par Interface-Z, nous n'avons pas besoin de modifier le composant OP correspondant au dispositif de liaison.
Nous avons essayé expérimentalement ces alternatives, et nous avons observé que presser permettait difficilement de contrôler la photo courante, alors que tourner le permettait plus facilement. Cette expérience montre que l'exploration abstraite de l'espace de conception est centrale mais ne remplace pas le prototypage et la conception pratique.

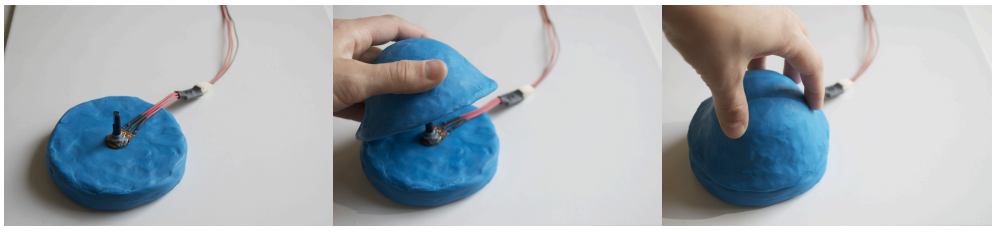


Figure 226: Prototypage de l'outil avec une modalité de liaison captant la rotation.

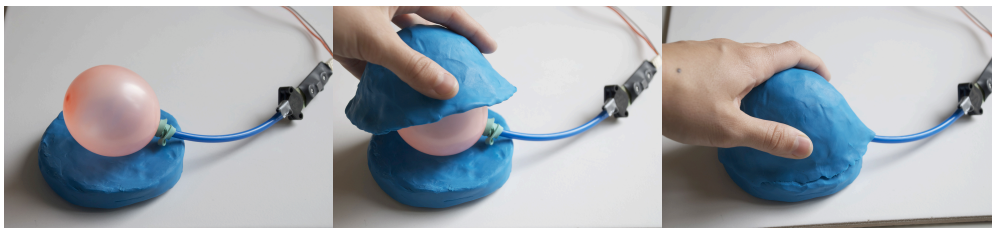


Figure 227: Prototypage de l'outil avec une modalité de liaison captant la pression.

L'espace de conception pour le langage de liaison est restreint par le choix du potentiomètre comme dispositif de liaison, et la propriété numérique *angle* (un entier dans un intervalle) nécessaire pour la tâche naviguer. Tout en répondant à ces contraintes, l'espace de conception nous a permis

d'explorer deux solutions, prototypées à la Figure 228 et à la Figure 229. À la Figure 228 avec le composant OP *IdentityInputLanguage*, il est nécessaire de faire tourner l'outil sur 360 degrés pour naviguer dans toutes les photos (langage de liaison global, section 2.1.1.2 du Chapitre 4). Comme faire tourner l'outil sur 360 degrés n'est pas facile à faire en un seul geste, nous avons imaginé un langage partiel et déformé (section 2.1.1.2 du Chapitre 4) : le système peut ne prendre qu'un sous-intervalle de rotation, faisable en un seul geste. La Figure 229 présente le code OP de ce langage de liaison utilisant le composant OP *RampInputLanguage*. Changer d'une solution vers l'autre demande de changer 3 lignes de code : la déclaration du composant (Figure 228 et Figure 229) et ses connexions, puisque le nom du composant change.

```
IdentityInputLanguage identity("identity", isOpposite, min, max);
```

Figure 228: Prototypage de l'outil avec un langage identité pour la liaison en entrée.

```
int lowerThreshold = 42;
int upperThreshold = 84;
RampInputLanguage ramp("ramp", isOpposite, min, lowerThreshold, upperThreshold, max);
```

Figure 229: Prototypage de l'outil avec un langage rampe pour la liaison en entrée.

En testant ces deux prototypes, nous avons remarqué que même si le langage de liaison *RampInputLanguage* permettait de facilement naviguer dans les photos en un unique mouvement, il n'était pas facile pour l'utilisateur d'ORBIS de comprendre pourquoi la totalité des 360 degrés n'étaient pas pris en compte. Nous avons donc choisi le langage *IdentityInputLanguage* de la Figure 228, même si les utilisateurs doivent faire plusieurs mouvements pour naviguer dans les photos.

Pour matérialiser la propriété numérique *level*, qui fournit un retour d'information, nous avons conçu et réalisé des ébauches de modalités de liaison en sortie avec OP. Nous voulions que ce retour d'information soit périphérique, c'est-à-dire qu'il ne distraie pas l'utilisateur de son point d'intérêt : la liste de photos sur l'écran. Nous avons exploré des dispositifs de liaison visuels (LED) ou sonores (haut-parleur), associés au langage de liaison *BeepOutputLanguage*. Nous avons aussi envisagé d'afficher un message par-dessus les photos sur l'écran ou matérialiser *level* via un servomoteur. Nous avons prototypé deux alternatives : l'une avec le numéro de la photo qui s'affiche pendant 1 seconde (*1000* millisecondes, Figure 230), et l'autre avec un haut-parleur (Figure 231). Le haut-parleur ou l'écran sont des dispositifs standards, et n'ont donc pas de composant OP. Nous n'avons pas pu prototyper la modalité de liaison avec la diode ni avec le servomoteur, car nous n'avions ni le matériel ni le composant dans cette première version de OP.

```
ShortDisplayOutputLanguage Message("Message", 1000, QVariant::String);
QObject::connect(&MyTurnableObjectDP, SIGNAL(LevelUpdated(QVariant)),
                 &Message, SLOT(Show(QVariant)));
```

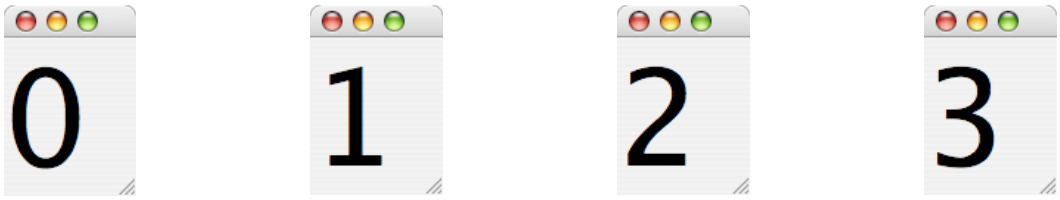


Figure 230 : Prototypage de la modalité de liaison en sortie avec un message qui s'affiche à l'écran : code OP (haut) et fenêtres affichées pendant 1 seconde par dessus l'objet de la tâche (bas, de gauche à droite) lorsqu'on affiche les photos du prototype de la première (à gauche) à la quatrième et dernière (à droite).

```
char* soundFile = "./Tink.aiff";
BeepOutputLanguage TurnBeepOL("TurnBeepOL", soundFile);
QObject::connect(&MyTurnableObjectDP, SIGNAL(LevelUpdated(int)),
                 &TurnBeepOL, SLOT(update()));
```

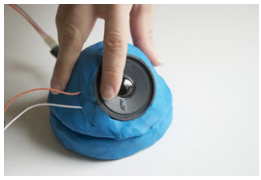


Figure 231: Prototypage de la modalité de liaison en sortie avec un haut-parleur : code OP (gauche) et haut parleur utilisé inséré dans le prototype de l'outil (droite).

Entre ces deux prototypes, seules quelques lignes changent, ce qui nous a permis d'essayer rapidement les deux solutions.

2.2.1.1.2. Exploration pratique des possibilités pour l'outil pour démarrer/arrêter la présentation

Pour l'outil de démarrage/arrêt de la présentation des photos, nous avons prototypé des alternatives avec capteur de flexion (Figure 232) ou de pression (Figure 233) d'Interface-Z. Pour ces deux alternatives qui utilisent le même type de ressource matérielle, le code OP est le même (lignes 1, 2 et 3 de la Figure 234).

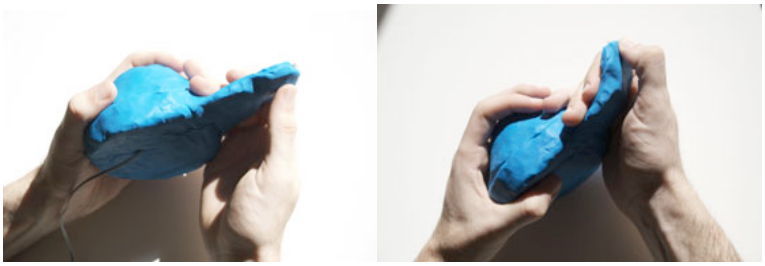


Figure 232 : Prototype physique de l'outil de démarrage/arrêt de la présentation avec une modalité de liaison captant la flexion.

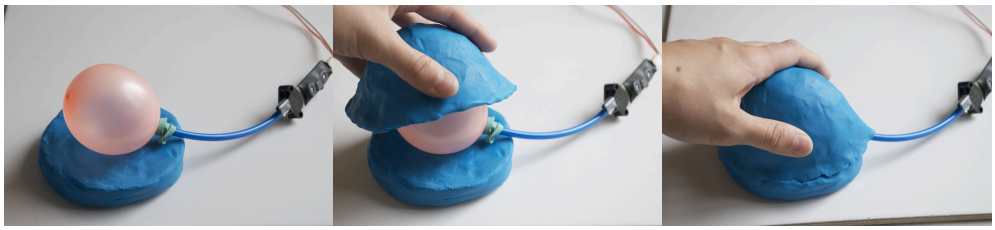


Figure 233 : Prototype physique de l'outil de démarrage/arrêt de la présentation avec une modalité de liaison captant la pression.

```
1  int indexOnBoard = 2;
2  bool isAnalogue = true;
3  int resolution = 7;
4  MIDIDevice Sensor("PressureSensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);
5
6  int thresholdValue = 50;
7  bool isTrueAbove = false;
8  ThresholdInputLanguage IL("Threshold", thresholdValue, isTrueAbove);
9
10 SqueezableObjectDigitalProperties MySqueezableObjectDP("MySqueezableObject");
11
12 char* soundFile = "./Pop.aiff";
13 BeepOutputLanguage PlayBeepOL("Beep", soundFile);
14
15 int duration = 1000;
16 QVariant::Type displayType = QVariant::Image;
17 ShortDisplayOutputLanguage SqueezeShortDisplayOL("ShortDisplayOL", duration, displayType);
18
19 QObject::connect(&Sensor, SIGNAL(updated(int, QTime)),
20                  &IL, SLOT(update(int, QTime)));
21 QObject::connect(&IL, SIGNAL(updated(bool, QTime)),
22                  &MySqueezableObjectDP, SLOT(updateIsPressed(bool)));
23 QObject::connect(&MySqueezableObjectDP, SIGNAL(IsPressedUpdated(bool)),
24                  &PlayBeepOL, SLOT(update(bool)));
25 QObject::connect(&MySqueezableObjectDP, SIGNAL(ImageUpdated(QImage*)),
26                  &SqueezeShortDisplayOL, SLOT(Show(QImage*)));
```

Figure 234 : Code OP de l'outil pour démarrer/arrêter la présentation des photos.

Là encore, nous utilisons des propriétés numériques sur mesure (Figure 234, ligne 10) développées avant l'expérimentation. Ce composant contient une propriété numérique booléenne *isPressed*.
Nous n'avions que le composant *ThresholdInputLanguage* (Figure 234, ligne 8) comme possibilité pour lier le dispositif *Sensor* (Figure 234, ligne 4) à la propriété numérique de type booléen (Figure 234, ligne 10) : il fallait que la sortie du composant langage de liaison soit booléenne et *ThresholdInputLanguage* était le seul candidat dans cette version d'OP. Nous avons pu facilement essayer expérimentalement plusieurs valeurs de seuil en changeant le paramètre *thresholdValue* (Figure 234, ligne 6).
Pour la sortie, nous avons prototypé avec les mêmes types de langages de liaison que pour l'outil précédent (Figure 234 : *BeepOutputLanguage* ligne 13 et *ShortDisplayOutputLanguage* ligne 17). En effet, le choix de composants était restreint pour cette première expérience. En revanche pour cet outil, nous avons choisi de conserver les deux alternatives et nous avons adapté le son associé au composant *BeepOutputLanguage* (Figure 234, ligne 12) : pour cet outil, c'est *Pop.aiff* qui sera utilisé, alors que c'était *Tink.aiff* pour l'outil précédent. La Figure 235 montre les images qui sont affichées par la deuxième réaction complémentaire au son, via le composant *ShortDisplayOutputLanguage* (Figure 234, ligne 17).



Figure 235 : Images affichées lorsque la présentation des photos est démarrée (gauche) ou arrêtée (droite).

2.2.1.1.3. Exploration pratique des possibilités pour l'outil pour mélanger les photos

Pour l'outil pour mélanger les photos, nous avons réalisé des prototypes avec un accéléromètre (Figure 236, ligne 4) comme dispositif d'entrée, puisque nous voulions secouer l'objet pour avoir

une métaphore de verbe (Chapitre 4, section 2.2.5.2) : « je secoue pour mélanger comme je ferais pour mélanger des dés ».

```
1  int indexOnBoard = 6;
2  bool isAnalogue = true;
3  int resolution = 7;
4  MIDIDevice ShakeSensor1("accelerometer1", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);
5
6  int threshold = 100;
7  int isTrueAbove = true;
8  int repeats = 3;
9  int interval = 3.0;
10 RepeatInputLanguage ShakeIL("ShakeIL", threshold, isTrueAbove, repeats, interval);
11
12 ShakableObjectDigitalProperties MyShakableObjectDP("MyShakableObject");
13
14 char* soundFile = "./Submarine.aiff";
15 BeepOutputLanguage ShuffleBeepOL("ShuffleBeepOL", soundFile);
16
17 int duration = 5000;
18 QVariant::Type displayType = QVariant::Image;
19 ShortDisplayOutputLanguage ShakeShortDisplayOL("ShortDisplayOL", duration, displayType);
20
21 QObject::connect(&ShakeSensor1, SIGNAL(updated(int, QTime)),
22                  &ShakeIL, SLOT(update(int, QTime)));
23 QObject::connect(&ShakeIL, SIGNAL(updated(bool, QTime)),
24                  &MyShakableObjectDP, SLOT(updateIsShaken(bool)));
25 QObject::connect(&MyShakableObjectDP, SIGNAL(IsShakenUpdated(bool)),
26                  &ShuffleBeepOL, SLOT(update(bool)));
27 QObject::connect(&MyShakableObjectDP, SIGNAL(ImageUpdated(QImage*)),
28                  &ShakeShortDisplayOL, SLOT(Show(QImage*)));
```

Figure 236 : Code OP de l'outil pour mélanger les photos.

Pour cet outil, nous avons développé un composant *RepeatInputLanguage* sur mesure (Figure 236, ligne 10), dont la sortie renvoyait vrai/faux selon si son entrée passe le seuil *threshold* (Figure 236, ligne 6) un certain nombre de fois (*repeats*, Figure 236, ligne 8) pendant un intervalle (*interval*, Figure 236, ligne 9). Nous constatons, pour ce cas, un développement à façon qui a été fait et donc une mise en œuvre du prototypage qui n'a pas été immédiate. De plus, nous n'avons pas exploré d'autres alternatives en ce qui concerne le langage de liaison en entrée.
De même, les propriétés numériques (Figure 236, ligne 12) sont sur mesure. Ces propriétés contiennent une propriété *isShaken* booléenne.
Pour la rendre observable, nous avons exploité les composants langages de sortie que nous avions à notre disposition : *BeepOutputLanguage* pour un retour sonore et *ShortDisplayOutputLangauge* pour un retour visuel (plus long 5 secondes que les précédents 3 secondes pour pouvoir le voir après avoir secoué). La Figure 237 présente ce retour visuel affiché par-dessus l'objet de la tâche.



Figure 237 : Image affichée lorsque l'outil dédié au mélange des photos est secoué (« RDM » signifie *Random*[51]).

[51] Aléatoire

2.2.1.2. Synthèse de l'utilisation de OP sur l'expérience d'ORBIS

Pendant cette expérience de prototypage et de conception avec ORBIS, nous avons pu mettre à l'épreuve la flexibilité de l'outil OP. Nous avons réagi au changement de conception des modalités de liaison, si les composants OP existaient. Nous avons aussi atteint les limites de l'outil OP pendant le prototypage de ORBIS, car nous n'avons pas pu prototyper beaucoup d'alternatives. Nous avons utilisé les composants simples existants (*IdentityInputLanguage*, *RampInputLanguage*, *BeepOutputLanguage*, *ShortDisplayOutputLanguage*). Clairement l'outil OP comportait peu de composants ce qui a été évidemment un frein au prototypage. Par exemple nous n'avons pas pu utiliser de diodes électroluminescentes. À l'issue de cette expérience, nous avons donc ajouté les composants *PhidgetInterfaceKitDevice*, *ARVideoInputDevice* et *ARTrackingInputLanguage* pour augmenter les possibilités offertes par l'outil et donc son pouvoir d'expression (puissance ou « plafond » dans les termes de [Myers et al., 2000]).
De plus, nous avons rencontré des difficultés à cause du manque de généricité. Certains langages ont été développés sur mesure (*RepeatInputLanguage*). De même, les propriétés numériques ont été définies pour l'application. Aussi il n'a pas été possible d'ajouter des propriétés numériques aux outils à la volée. Nous avons décidé à l'issue de cette expérience de les décomposer en éléments plus génériques réutilisables, comme préconisé par [Klemmer et al., 2004].
Nous avons également été dans l'impossibilité de simuler des dispositifs que nous ne possédions pas, car nous n'avions pas encore d'interface graphique. Nous en avons donc développé une première version avant la seconde expérience.
Pour cette évaluation, l'utilisateur de l'outil OP était un expert (auteur de l'outil) et le travail de conception a été effectué en binôme avec un designer (arts appliqués). Le retour que nous avons eu sur les prototypes de la part du designer ont été positifs : les prototypes finaux ont été présentés devant un jury en arts appliqués , et le designer nous a dit avoir apprécié les prototypes réalisés. En effet, il avait toujours eu l'habitude de prototyper l'apparence de ses objets sans l'interaction.

Nous présentons maintenant notre seconde expérience, sur le projet de conception de Roam, qui a fait intervenir un autre designer que celui pour ORBIS.

2.2.2. Étude expérimentale pour le système Roam

Roam[52] est un système mobile d'enregistrement sonore et photographique alternatif, qui ne distrait pas l'utilisateur de ce qui l'entoure, au contraire des appareils photos traditionnels. Avec un appareil photo, l'utilisateur concentre son attention sur le système plutôt que sur le monde qui l'entoure. Au contraire, *Roam* vise à rester au second plan de l'attention de l'utilisateur et ne pas le distraire de ce qui l'intéresse.
Nous avons conçu avec un designer (Chapitre 5, section 3.1.2.2) un outil pour enregistrer le son. L'enregistrement sonore (l'objet de la tâche) a aussi été prototypé. Nous ne le décrivons pas ici, car il est strictement numérique (il n'est pas possible de le percevoir l'enregistrement sonore en utilisant *Roam*). Comme pour ORBIS, l'évaluation ne concernait pas la difficulté de prise en main de l'outil car un des deux utilisateurs était expert. Nous détaillons ici les prototypes que nous avons réalisés pour l'outil d'enregistrement sonore.

2.2.2.1. Description

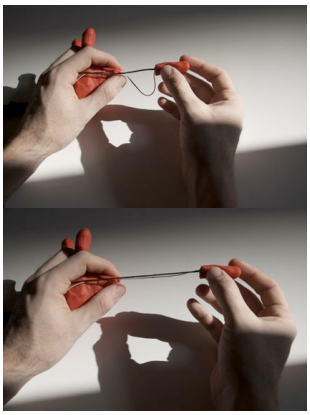
Nous avons utilisé le composant générique pour les propriétés numériques *isOn* (Figure 238, ligne 10) et *isOk* (Figure 238, ligne 12). Nous précisons en second paramètre que ces propriétés sont de type booléen. Ce nouveau composant nous a permis de rajouter à la volée le composant pour la propriété *isOk*, en quelques secondes pendant la conception. Ceci nous aurait pris plusieurs minutes pendant l'expérience précédente.

[52] vadrouiller

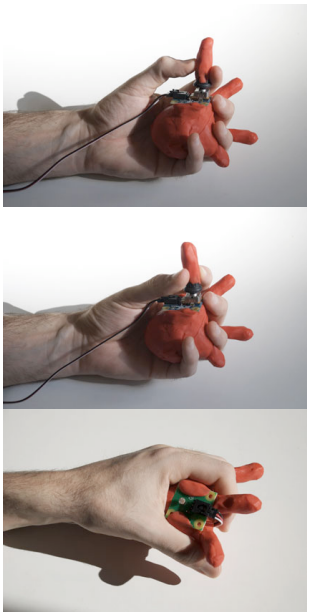
```
1  int indexOnBoard = 0;
2  bool isAnalogue = true;
3  int resolution = 7;
4  MIDIDevice Sensor("Sensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);
5
6  int threshold = 50;
7  bool isTrueAbove = true;
8  ThresholdInputLanguage Threshold("Threshold", threshold, isTrueAbove);
9
10 DigitalProperty isOn("isOn", QVariant::Bool);
11
12 DigitalProperty isOk("isOk", QVariant::Bool);
13
14 BeepOutputLanguage BeepOLOn("BeepOLOn", "");
15
16 BeepOutputLanguage BeepOLOk("BeepOLOk", "");
17
18 indexOnBoard = 0;
19 isAnalogue = false;
20 PhidgetInterfaceKitDevice PhidgetLEDgreen("PhidgetLEDgreen", LinkingComponent::OUT, indexOnBoard, isAnalogue);
21
22 indexOnBoard = 1;
23 PhidgetInterfaceKitDevice PhidgetLEDyellow("PhidgetLEDyellow", LinkingComponent::OUT, indexOnBoard, isAnalogue);
24
25 QObject::connect(&Sensor, SIGNAL(updated(int, QTime)),
26                  &Threshold, SLOT(update(int, QTime)));
27 QObject::connect(&Threshold, SIGNAL(updated(QVariant, QTime)),
28                  &isOn, SLOT(updateProperty(QVariant)));
29 QObject::connect(&isOn, SIGNAL(PropertyUpdated(QVariant)),
30                  &BeepOLOn, SLOT(update(void)));
31 QObject::connect(&isOk, SIGNAL(PropertyUpdated(QVariant)),
32                  &BeepOLOk, SLOT(update(void)));
33 QObject::connect(&BeepOLOn, SIGNAL(updated(bool)),
34                  &PhidgetLEDgreen, SLOT(UpdateOutput(bool)));
35 QObject::connect(&BeepOLOk, SIGNAL(updated(bool)),
36                  &PhidgetLEDyellow, SLOT(UpdateOutput(bool)));
```

Figure 238 : Code OP complet de l'outil d'enregistrement pour Roam.

Pour cet outil, nous avons essayé plusieurs modalités de liaison en entrée. Nous avons tout d'abord essayé plusieurs dispositifs : capteur de flexion (Figure 244), d'étirement (Figure 239, haut), potentiomètre (Figure 239, centre), capteur de luminosité (Figure 239, bas). Entre les différentes alternatives logicielles prototypées avec OP présentées à la Figure 238 et à la Figure 239, il n'y a que quelques lignes qui sont modifiées.
Nous avons utilisé *ThresholdInputLanguage* (Figure 238, ligne 8) comme langage de liaison en entrée, connecté (Figure 238, lignes 27 et 28) vers une propriété numérique *isOn* booléenne (Figure 238, ligne 10). Nous pouvons expérimenter des alternatives rapidement modifiant le seuil et le fait qu'il faille aller sur ou sous le seuil en changeant les valeurs des variables *threshold* et *isTrueAbove* lignes 6 et 7 de la Figure 238. Nous aurions pu utiliser *RepeatInputLanguage* pour prototyper facilement d'autres alternatives de liaison en entrée, mais nous avons conceptuellement eliminé cette solution pour ne pas fatiguer la main de l'utilisateur.



```
int indexOnBoard = 0;
bool isAnalogue = true;
int resolution = 7;
MIDIDevice Sensor("Sensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);
```



```
int indexOnBoard = 0;
bool isAnalogue = true;
PhidgetInterfaceKitDevice Sensor("Sensor", LinkingComponent::IN, indexOnBoard, isAnalogue);
```

```
int indexOnBoard = 0;
bool isAnalogue = true;
PhidgetInterfaceKitDevice Sensor("Sensor", LinkingComponent::IN, indexOnBoard, isAnalogue);
```

Figure 239 : Alternatives physiques et logicielles pour les dispositifs de liaison en entrée pour l'outil d'enregistrement de Roam : capteur d'étirement d'Interface-Z (haut), potentiomètre des Phidgets (centre), et capteur de luminosité des Phidgets (bas).

Pour cet outil, nous avons essayé plusieurs modalités de liaison en sortie pour rendre perceptible les propriétés *isOn* (Figure 238, ligne 10) et *isOk* (Figure 238, ligne 12). Nous avons tout d'abord pensé à plusieurs dispositifs disponibles : un haut-parleur (Figure 240, gauche) ou des diodes (Figure 240, droite).



Figure 240: Alternatives de conception : sortie sonore avec un haut-parleur (à gauche), lumineuse avec une LED (à droite).

Pour ces dispositifs, nous avons essayé plusieurs langages disponibles dans les composants OP : Nous avons testé *BeepOutputLanguage* avec ou sans répétition grâce au composant *RepeatLanguage*. La Figure 241 présente les codes OP pour *isOn* et *isOk* avec un *BeepOutputLanguage* sans répétition et avec le haut-parleur comme dispositif (cas de la Figure 240, à gauche). La Figure 242 présente le code OP pour *BeepOutputLanguage* sans répétition et avec la diode comme dispositif (cas de la Figure 240, à droite). La Figure 243 présente le code OP pour *BeepOutputLanguage* avec répétition dans le cas de l'utilisation avec la diode : à la Figure 243 nous appliquons cette alternative à la propriété numérique *isOn*. Par rapport à l'alternative précédente, il nous suffit d'intercaler un composant *RepeatLanguage* entre la propriété et le composant *BeepOutputLanguage*.

```
char* soundFile = "./Tink.aiff";
BeepOutputLanguage BeepOLOn("BeepOLOn", soundFile);
QObject::connect(&isOn, SIGNAL(PropertyUpdated(QVariant)),
                 &BeepOLOn, SLOT(update(void)));
```

```
char* soundFile = "./Submarine.aiff";
BeepOutputLanguage BeepOLOk("BeepOLOk", soundFile);
QObject::connect(&isOk, SIGNAL(PropertyUpdated(QVariant)),
                 &BeepOLOk, SLOT(update(void)));
```

Figure 241 : Alternatives utilisant le composant *BeepOutputLanguage* avec le haut-parleur pour l'outil d'enregistrement de Roam.

```
BeepOutputLanguage BeepOLOn("BeepOLOn", "");

BeepOutputLanguage BeepOLOk("BeepOLOk", "");

indexOnBoard = 0;
isAnalogue = false;
PhidgetInterfaceKitDevice PhidgetLEDgreen("PhidgetLEDgreen", LinkingComponent::OUT, indexOnBoard, isAnalogue);

indexOnBoard = 1;
PhidgetInterfaceKitDevice PhidgetLEDyellow("PhidgetLEDyellow", LinkingComponent::OUT, indexOnBoard, isAnalogue);

QObject::connect(&isOn, SIGNAL(PropertyUpdated(QVariant)),
                 &BeepOLOn, SLOT(update(void)));
QObject::connect(&isOk, SIGNAL(PropertyUpdated(QVariant)),
                 &BeepOLOk, SLOT(update(void)));
QObject::connect(&BeepOLOn, SIGNAL(updated(bool)),
                 &PhidgetLEDgreen, SLOT(UpdateOutput(bool)));
QObject::connect(&BeepOLOk, SIGNAL(updated(bool)),
                 &PhidgetLEDyellow, SLOT(UpdateOutput(bool)));
```

Figure 242 : Alternative utilisant le composant *BeepOutputLanguage* avec la diode pour l'outil d'enregistrement de Roam, afin de matérialiser les propriétés *isOn* et *isOk*.

```
int repeats = 3;
float interval = 0.5;
RepeatLanguage RLOn("Repeat", LinkingComponent::OUT, repeats, interval);

BeepOutputLanguage BeepOLOn("BeepOLOn", "");

indexOnBoard = 0;
isAnalogue = false;
PhidgetInterfaceKitDevice PhidgetLEDgreen("PhidgetLEDgreen", LinkingComponent::OUT, indexOnBoard, isAnalogue);

QObject::connect(&isOn, SIGNAL(PropertyUpdated(QVariant)),
                 &RLOn, SLOT(update(QVariant)));
QObject::connect(&RLOn, SIGNAL(updated(bool)),
                 &BeepOLOn, SLOT(update()));
QObject::connect(&BeepOLOn, SIGNAL(updated(bool)),
                 &PhidgetLEDgreen, SLOT(UpdateOutput(bool)));
```

Figure 243 : Alternatives utilisant le composant *BeepOutputLanguage* avec une répétition (composant *RepeatLanguage*) avec la diode pour la propriété *isOn* de l'outil d'enregistrement de Roam.

Nous avons aussi pensé à une réaction de l'outil plus compliquée avec plusieurs diodes vertes qui s'allument les unes après les autres. Nous n'avons pas pu le prototyper immédiatement car le composant *DelayLanguage* n'existait pas encore. Nous l'avons rajouté ensuite.
La Figure 244 présente le code OP correspondant au prototype logiciel complet de la Figure 238.



Figure 244: Prototype physique complet de l'outil dans Roam (correspondant au code OP de la Figure 238) : dans la main de l'utilisateur (gauche) et en gros plan (droite), avec capteur de flexion et diodes électroluminescentes, une verte et une jaune.

Nous présentons maintenant nos conclusions sur cette expérience de prototypage avec Roam.

2.2.2.2. Synthèse de l'utilisation de OP sur l'expérience de Roam

Durant cette expérience, notre activité de prototypage a été plus riche et nous avons pu prototyper plus d'alternatives, grâce à la généricité de *RepeatLanguage* et des propriétés numériques, ainsi qu'à l'ajout des dispositifs Phidgets.
Comme dans le cas d'ORBIS, certaines solutions étaient difficilement réalisables avec les composants OP disponibles. Nous avons donc à nouveau enrichi l'ensemble des composants OP notamment avec un composant *DelayLanguage* et des dispositifs de sorties Interface-Z (comme les servomoteurs). De plus, construire l'interface graphique pendant la conception nous a paru trop long (quelques minutes). Nous avons donc amélioré le constructeur de l'interface graphique.
Nous avons aussi appris avec cette expérience que pour faire participer activement le designer au prototypage, il nous faudra concevoir une interface graphique à notre outil adapté à ces futurs utilisateurs.

Nous avons présenté deux expériences en situation réelle, effectuées en collaboration avec des designers. Ces deux expériences nous ont permis d'enrichir l'ensemble des composants OP. Pour ces expériences, nous avons utilisé OP dans un contexte réel, afin de voir si l'outil était adapté à ces situations de prototypage. En revanche, nous n'avons pas encore évalué son utilisation par des utilisateurs non-experts. Pour cela, nous avons réalisé des études expérimentales en laboratoire, que nous présentons maintenant.

2.3. Évaluation expérimentale en laboratoire de la boîte à outils OP

Notre objectif lors de cette expérience est d'évaluer l'utilisation de la boîte à outils OP par des utilisateurs informaticiens, mais sans expérience de l'outil OP. Cette évaluation constitue une première étape vers l'évaluation de la boîte à outils par des utilisateurs novices. Nous présentons d'abord le protocole choisi, puis les expériences que nous avons effectuées.

2.3.1. Protocole

L'objectif de l'évaluation est d'obtenir des informations sur le lien entre le modèle d'interaction mixte et l'outil OP, ainsi que sur l'outil OP lui-même. Nous décrivons ici le protocole choisi pour le déroulement des sessions d'expérience, ainsi que le protocole retenu pour recruter les participants à l'expérience.

2.3.1.1. Sessions d'expérience

Le protocole que nous avons retenu est itératif : nous l'avons amélioré entre les sessions. Notre objectif n'était pas de comparer ni de faire une évaluation quantitative, mais d'obtenir des retours sur notre outil de prototypage. Nous présentons les éléments de l'expérience qui ont été soumis aux itérations ci-dessous lors la description du déroulement des expériences.
Afin de favoriser le débat, nous avons choisi de mener les évaluations sous forme de groupes focalisés avec 4 participants. Le protocole favorisait le débat et l'échange autour de l'outil, car nous voulions recueillir des critiques constructives de la part des participants. La durée prévue des sessions était de 2 heures. Les sessions étaient enregistrées sous forme audio et vidéo. De plus, comme nous l'avons souligné à la section 1.2.2.3, nous avons cherché à éviter de poser des questions déjà orientées, par exemple avec la formulation « raccourcir le temps » au lieu de demander si « le temps requis pour accomplir une tâche est (1) plus long ou (2) moins long ».
Pendant l'évaluation, nous avons :
observé les participants,

1. recueilli leurs réponses écrites,

2. fait présenter leurs réponses aux participants,
3. fait débattre les participants pour recueillir leurs remarques et suggestions.

Le Tableau 28 présente le déroulement prévu des sessions d'évaluation que nous décrivons maintenant.

Heure	*Durée*	*Activité*
0h00 0h05	5 min	**Introduction**
0h05 0h25	20 min	**Exercices 1 puis 2**
0h25 0h55	30 min	**Discussion**
0h55 1h30	35 min	**Modifications de code OP**
1h30 2h00	30 min	**Discussion**

Tableau 28 : Déroulement prévu des sessions d'évaluation en groupe focalisé.

L'introduction des sessions consiste à faire connaissance : chacun se présente, puis répond à un questionnaire sur son travail. D'une session à l'autre, selon l'expertise des participants, nous leur avons posé des questions complémentaires sur l'expertise qu'ils avaient (par exemple : « quelle est la principale application que vous avez développée avec Qt ? »). Ces questionnaires sont présentés en Annexe E.

L'objectif de la première partie de l'expérience était d'évaluer le lien entre la description de l'interaction selon notre modèle et l'outil de prototypage. Pour cela, nous avons préparé deux exercices.
Le premier demandait aux participants de comprendre une description de l'interaction et d'écrire un pseudo code à partir de cette description. Nous leur demandions ensuite comment ils font habituellement (le cas échéant), puis comment il aimerait coder ce problème idéalement. Les participants tiraient au sort le type de description sur lequel ils travaillaient : deux d'entre eux ont travaillé sur un scénario textuel, les deux autres sur une description avec le modèle d'interaction mixte.
Le second exercice demandait aux participants de comprendre un programme, prototype logiciel, réalisé avec l'outil de prototypage évalué. Les outils de prototypage que nous voulions évaluer sont OP et les Phidgets. Les participants doivent décrire l'interaction correspondant à ces lignes de code. Puis nous leur demandons d'imaginer une architecture logicielle adaptée à ce type de développement.
Ils disposaient de 10 minutes pour faire chaque exercice, et ne recevaient aucune aide ou explication sur les outils évalués. Une fois leurs réponses rendues, nous leur demandions de présenter leurs réponses aux deux exercices, en commençant par les participants ayant travaillé avec la description sous forme de texte, afin que les autres ne les influencent pas. Cette phase de discussion durait environ ½ heure et nous a permis d'avoir leurs retours sur le lien entre la description de l'interaction et l'outil de prototypage.

L'objectif de la deuxième partie de l'expérience était d'évaluer l'utilisation concrète de la boîte à outils par les participants. Pour cela, nous leur demandions de faire des modifications que nous avons rencontrées lors de prototypage réel (avec l'expérience d'ORBIS). Ces exercices de modifications ont été réalisés deux par deux, en mélangeant dans chaque binôme un sujet ayant eu une description sous forme de texte avec un sujet ayant eu une description avec le modèle d'interaction mixte. Ils disposaient de 10 minutes pour faire chaque modification. Ils pouvaient consulter la documentation de l'outil, mais ne recevaient aucune autre aide ou explication sur l'outil évalué. En revanche, l'expérimentateur répondait à leurs questions concernant la plateforme utilisée (demande de raccourcis claviers par exemple, questions relatives à XCode, etc.), car ces outils n'étaient pas évalués.

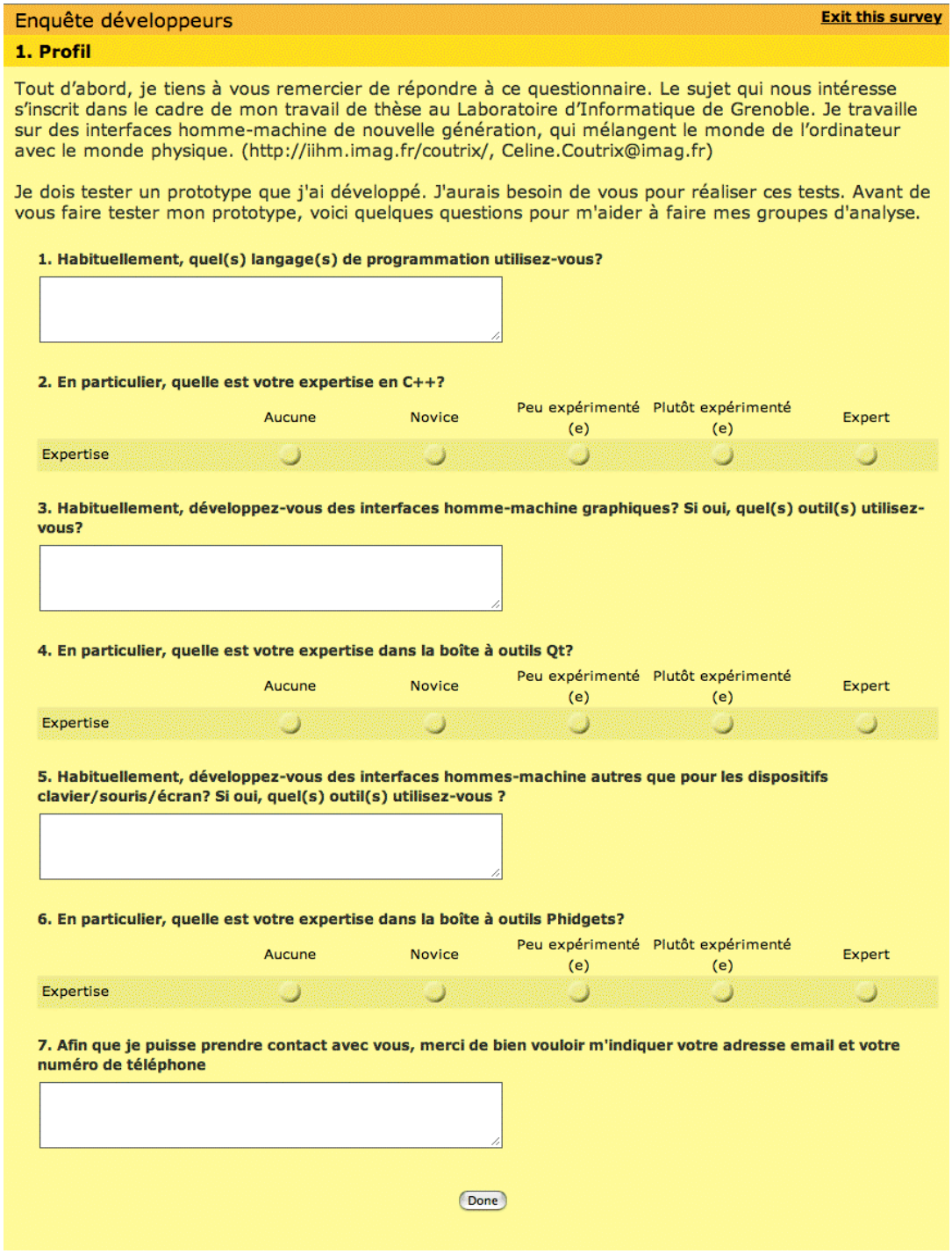
Nous voulions réaliser ces expériences avec OP et les Phidgets, ainsi qu'avec différents profils d'utilisateur (Tableau 29) : des informaticiens novices dans les outils à manipuler (C++, Qt, OP et les Phidgets), des informaticiens connaissant Qt, et des informaticiens connaissant les Phidgets. Ces choix nous demandaient de recruter les ressources humaines décrites au Tableau 29. Pour recruter ces participants, nous avons mis en place le protocole que nous décrivons maintenant.

Outil	OP		Phidgets	
Description	Texte	Modèle d'interaction mixte	Texte	Modèle d'interaction mixte
Informaticiens novices dans les outils à manipuler	2	2	2	2
Informaticiens connaissant les outils C++ et Qt	2	2	2	2
Informaticiens connaissant les Phidgets	2	2		

Tableau 29 : Sessions d'évaluation : nombre de participants pour chaque outil (OP ou Phidgets), description (Texte ou Modèle d'interaction mixte) et profils des utilisateurs prévus.

2.3.1.2. Recrutement des participants

Pour recruter les sujets participants aux expériences, nous avons lancé une enquête par courrier électronique auprès des personnels du Laboratoire d'Informatique de Grenoble (LIG). Ils étaient invités à répondre à un questionnaire sur le Web, dont nous présentons à la Figure 245 une capture d'écran.


Enquête développeurs

Exit this survey

1. Profil

Tout d'abord, je tiens à vous remercier de répondre à ce questionnaire. Le sujet qui nous intéresse s'inscrit dans le cadre de mon travail de thèse au Laboratoire d'Informatique de Grenoble. Je travaille sur des interfaces homme-machine de nouvelle génération, qui mélangent le monde de l'ordinateur avec le monde physique. (http://iihm.imag.fr/coutrix/, Celine.Coutrix@imag.fr)

Je dois tester un prototype que j'ai développé. J'aurais besoin de vous pour réaliser ces tests. Avant de vous faire tester mon prototype, voici quelques questions pour m'aider à faire mes groupes d'analyse.

1. Habituellement, quel(s) langage(s) de programmation utilisez-vous?

2. En particulier, quelle est votre expertise en C++?

	Aucune	Novice	Peu expérimenté (e)	Plutôt expérimenté (e)	Expert
Expertise					

3. Habituellement, développez-vous des interfaces homme-machine graphiques? Si oui, quel(s) outil(s) utilisez-vous?

4. En particulier, quelle est votre expertise dans la boîte à outils Qt?

	Aucune	Novice	Peu expérimenté (e)	Plutôt expérimenté (e)	Expert
Expertise					

5. Habituellement, développez-vous des interfaces hommes-machine autres que pour les dispositifs clavier/souris/écran? Si oui, quel(s) outil(s) utilisez-vous ?

6. En particulier, quelle est votre expertise dans la boîte à outils Phidgets?

	Aucune	Novice	Peu expérimenté (e)	Plutôt expérimenté (e)	Expert
Expertise					

7. Afin que je puisse prendre contact avec vous, merci de bien vouloir m'indiquer votre adresse email et votre numéro de téléphone

Done

Figure 245 : Capture d'écran de l'enquête auprès des personnels du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé.

Il est important de souligner que sur les soixante-dix personnes qui ont répondu, toutes ne sont pas informaticien(ne)s. À la première question, sur 70 réponses, 5 participants se sont déclarés non informaticiens. Les autres ont principalement dit utiliser Java (39 personnes), C++ (34 personnes), C (33 personnes). Nous présentons à la première ligne du Tableau 30 et à la Figure 246 les résultats de la question 2. Nous remarquons que beaucoup de participants connaissent C++, mais qu'ils existent suffisamment de participants non-experts (aucune connaissance ou novice) pour faire un groupe focalisé de non-experts exclusivement. Pour trouver des participants informaticiens novices, nous avons choisi 4 personnes au hasard parmi les informaticiens de ce groupe.

	Aucune	*Novice*	*Peu expérimenté(e)*	*Plutôt expérimenté(e)*	*Expert*
C++	8	5	20	29	7
Qt	50	9	6	2	2
Phidgets	65	1	1	0	0

Tableau 30 : Résultats des questions 2, 4 et 6 de l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : nombre de sujets selon leur expertise en C++, Qt et Phidgets.

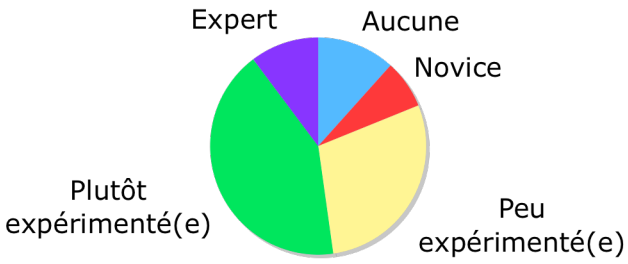


Figure 246 : Représentation graphique de l'expertise en C++ des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 8 n'ayant aucune connaissance, 5 novices, 20 peu expérimenté(e)s, 29 plutôt expérimenté(e)s, 7 experts.

D'après les résultats à la question 3 de l'enquête, nous remarquons que, sur 67 qui ont répondu à la question, 33 ne développent pas habituellement d'interfaces homme-machine graphiques. 25 personnes se sont notées de novices à expertes en Qt lors de cette enquête, comme le montrent le Tableau 30 et la Figure 248. Les outils les plus cités (hors Qt) sont : Java Swing (11 personnes), Gtk/Glade (4 personnes), Tcl/Tk (3 personnes), Quartz ou Cocoa (3 personnes), Glut ou OpenGL (3 personnes), Java AWT (2 personnes).
Pour le second groupe focalisé, nous avons choisi au hasard 4 participants dans le groupe des participants connaissant Qt (de novice à plutôt expérimenté(e)).

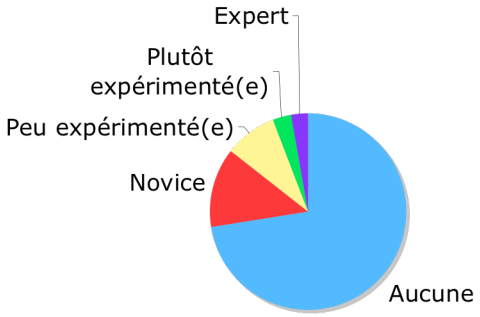


Figure 247 : Représentation graphique de l'expertise en Qt des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 50 n'ayant aucune connaissance, 9 novices, 6 peu expérimenté(e)s, 2 plutôt expérimenté(e)s, 2 experts.

Sur 57 personnes, 37 ont déclaré ne pas développer habituellement des interfaces homme-machine autres que pour les dispositifs clavier/souris/écran. Comme le montrent le Tableau 30 et la Figure 248, nous n'avons pas pu trouver suffisamment de sujets connaissant les Phidgets pour mener la comparaison avec cette boîte à outils. Dans la perspective d'une évaluation plus longue, nous avons comme projet de former des participants aux Phidgets comme à OP, avant de les faire participer à cette expérience. Nous n'avons donc pas encore réalisé d'expérience avec les Phidgets. Le Tableau 31 montre les sessions d'évaluation que nous avons réalisées.

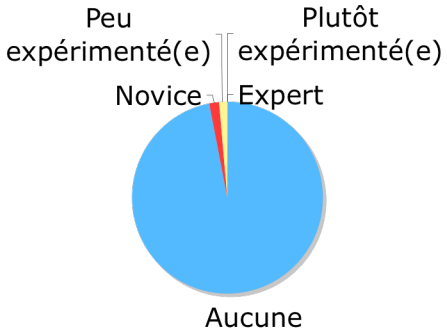


Figure 248 : Représentation graphique de l'expertise en Phidgets des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 65 n'ayant aucune connaissance, 1 novice, 1 peu expérimenté(e), aucun plutôt expérimenté(e), aucun expert.

Outil	OP	
Description	Texte	Modèle d'interaction mixte
Informaticiens novices dans les outils à manipuler	2	2
Informaticiens connaissant les outils C++ et Qt	2	2

Tableau 31 : Sessions d'évaluation réalisées : nombre de participants pour l'outil OP évalué selon la description (Texte ou Modèle d'interaction mixte) et le profil des utilisateurs.

Nous avons réalisé deux expériences en groupe focalisé, le premier avec des informaticiens novices aux outils (C++, Qt, OP) et le second familiarisé avec Qt, la boîte à outils sur laquelle est basée OP.

2.3.2. Première expérience

Cette expérience avait pour but d'évaluer l'utilisation d'OP et son lien avec le modèle d'interaction mixte avec des informaticiens novices en C++, Qt et Phidgets (peu ou aucune connaissance de ces outils).
La Figure 249 montre une image issue de la vidéo de cette expérience pilote. Deux participants échangent au premier plan et une expérimentatrice les observe à l'arrière-plan.



Figure 249: Expérience pilote avec des novices en C++, Qt et Phidgets (peu ou aucune connaissance de ces outils). Au premier plan, deux participants et à l'arrière-plan, une expérimentatrice.

Nous présentons d'abord les supports que nous avons fourni aux participants pour réaliser les exercices de cette première expérience, puis les résultats obtenus dans ces conditions.

2.3.2.1. Déroulement

La première partie comprend deux exercices. Pour le premier exercice, nous avons fourni aux participants les descriptions présentées à la Figure 250.

Marie manipule un objet sensible à la lumière. Il contient un capteur de luminosité et est muni d'une diode électroluminescente (LED) à sa surface. Lorsqu'elle approche trop cet objet d'une source lumineuse, la diode clignote une fois. Alors, Marie éloigne l'objet de la lumière : la diode clignote une nouvelle fois quand l'objet est suffisamment loin de la source lumineuse.

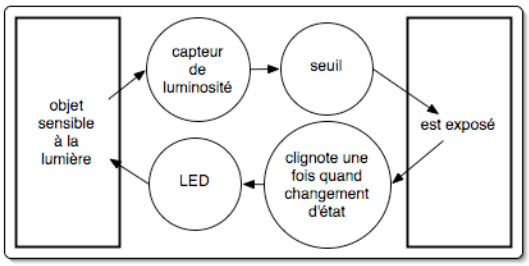


Figure 250 : Descriptions d'un même objet mixte fournies pour le premier exercice aux participants de la première expérience.

Pour le second exercice, nous avons fourni aux participants le code OP présenté à la Figure 251.

```
int SliderIndex = 0;
bool isSliderAnalogue = true;
int LEDIndex = 0;
bool isLEDAnalogue = false;
int ValueMin = 0;
int ValueMax = 999;

PhidgetInterfaceKitDevice Slider("Slider", LinkingComponent::IN, SliderIndex, isSliderAnalogue);
IdentityInputLanguage IL("IL", false, ValueMin, ValueMax);
DigitalProperty Level("Level", QVariant::Int);
BeepOutputLanguage Beep("BeepOLOn", "");
PhidgetInterfaceKitDevice LED("PhidgetLEDgreen", LinkingComponent::OUT, LEDIndex, isLEDAnalogue);

QObject::connect(&Slider, SIGNAL(updated(int, QTime)),
                 &IL, SLOT(update(int, QTime)));
QObject::connect(&IL, SIGNAL(updated(QVariant, QTime)),
                 &Level, SLOT(updateProperty(QVariant)));
QObject::connect(&Level, SIGNAL(PropertyUpdated(QVariant)),
                 &Beep, SLOT(update(void)));
QObject::connect(&Beep, SIGNAL(updated(bool)),
                 &LED, SLOT(UpdateOutput(bool)));
```

Figure 251 : Code OP fourni aux participants du groupe focalisé pour le 2ème exercice.

La deuxième partie comprend des exercices pratiques de modifications de code sur machine. Les modifications à faire à partir du code présenté à la Figure 252, tirées au sort par les participants, sont présentées (dans l'ordre de réalisation) à la Figure 253 et à la Figure 254. Ces modifications, lors de cette expérience, étaient données telles quelles et n'étaient pas expliquées afin d'identifier les difficultés des participants.

```
int Index = 0;
bool isAnalogue = true;
int Resolution = 7;
MIDIDevice TurnSensor("TurnSensor", LinkingComponent::IN, Index, isAnalogue, Resolution);

bool isOpposite = false;
int Min = 0;
int LowerThreshold = 0;
int UpperThreshold = 127;
int Max = 127;
RampInputLanguage IdentityIL("IdentityIL", isOpposite, Min, LowerThreshold, UpperThreshold, Max);

QObject::connect(&TurnSensor, SIGNAL(updated(int, QTime)),
                 &IdentityIL, SLOT(update(int, QTime)));

DigitalProperty Angle("Angle", QVariant::Int);

QObject::connect(&IdentityIL, SIGNAL(updated(QVariant, QTime)),
                 &Angle, SLOT(updateProperty(QVariant)));

DigitalProperty Level("Level", QVariant::Int);

BeepOutputLanguage BeepOL("BeepOL", "");

QObject::connect(&Level, SIGNAL(PropertyUpdated(QVariant)),
                 &BeepOL, SLOT(update(void)));

Index = 0;
isAnalogue = false;
PhidgetInterfaceKitDevice PhidgetLED("PhidgetLED", LinkingComponent::OUT, Index, isAnalogue);

QObject::connect(&BeepOL, SIGNAL(updated(bool)),
                 &PhidgetLED, SLOT(UpdateOutput(bool)));
```

Figure 252 : Code OP à modifier.

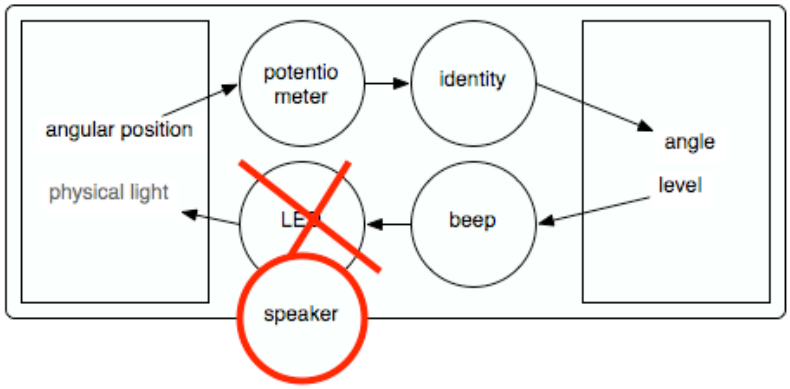


Figure 253 : Première modification à effectuer (Chaque modification étant tirée au sort par les participants).

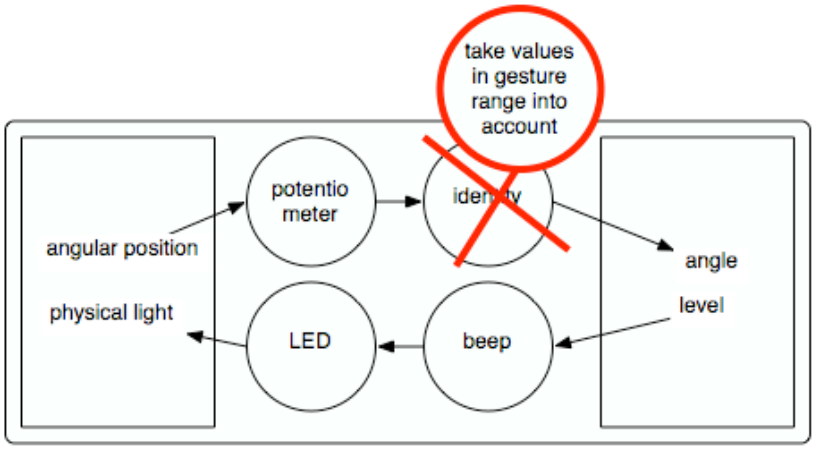


Figure 254 : Deuxième modification à effectuer (Chaque modification étant tirée au sort par les participants).

Nous présentons maintenant les résultats de cette première expérience de l'outil OP avec des novices.

2.3.2.2. Résultats

Nous avons noté à partir de l'enregistrement vidéo les points positifs et les limitations identifiés lors de cette première expérience.

2.3.2.2.1. Points positifs concernant le lien entre le modèle d'interaction mixte et l'outil de prototypage

De façon générale, nous avons remarqué que les participants qui ont eu la description sous forme de modèle (Figure 250, à droite) ont terminé d'écrire leur pseudo code avant les deux autres. Un des participants qui a eu le modèle a fait un schéma comportant des similarités avec celui qu'il avait eu avant pour expliquer le code (Figure 255). Nous avons aussi remarqué que ceux qui ont eu le schéma ont fini l'exercice d'explication de code en premier.

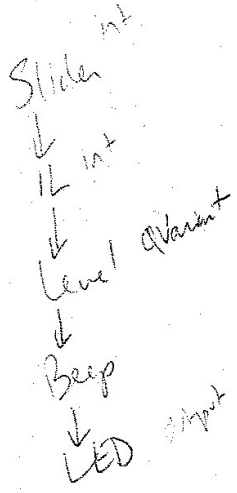


Figure 255 : Schéma réalisé par un participant de la première expérience pour expliquer le code de la Figure 251.

Les discussions nous permettent de voir plus clairement ce qu'ils retiennent comme intéressant. Parmi les choix que nous avons faits pour traduire l'outil de conception en outil de prototypage, certains se trouvent confirmés par des observations ou remarques.

Tout d'abord, le pseudo-code proposé par un participant contient une couche qui abstrait du niveau dispositif (classe *LED* et classe *SunSensor*) Ceci confirme en partie l'architecture de l'outil OP reprenant les concepts du modèle d'interaction mixte. De plus, le participant a proposé ceci bien qu'il ait eu le texte comme description. Ceci nous encourage à faire des évaluations de l'outil de prototypage sans son outil conceptuel afin de mesurer sa propre contribution.

Ensuite, nous avons étudié comment les participants faisait la correspondance entre les concepts manipulés dans le code et les concepts représentés dans le modèle d'interaction mixte (par exemple la correspondance entre les flèches dans le modèle d'interaction mixte et les lignes commençant par QObject ::connect dans le code).
De façon générale, les quatre participants ont déclaré faire un schéma avant de programmer. De plus, un participant, parlant du code du deuxième exercice (Figure 251) par rapport au schéma qu'il avait eu avant (Figure 250, à droite) dit que « ça ressemblait beaucoup ». Un autre participant, en parlant du code présenté à la Figure 251 dit qu'« on aurait plus envie de faire un schéma que de l'écrire […]. Je vois QObject::connect slider à IL, IL à Level, Level à Beep, Beep à LED; alors […] je fais un schéma avec mes objets, je les lie". De même, un troisième participant dit « la première chose quand on veut comprendre ça, on a envie de faire un schéma ». Il a réalisé sur sa feuille le schéma présenté à la Figure 255 à côté du code qu'il devait expliquer (Figure 251).
Nous constatons donc qu'à partir du code, les participants veulent élaborer un schéma. Les descriptions adoptées semblent confirmer notre représentation de l'interaction. Nous verrons les limites de cet aspect dans la section suivante.
De même, pendant la discussion après l'exercice de modifications où les participants avaient à la fois un schéma (Figure 253 et Figure 254) et le code OP correspondant (Figure 252), un des participants dit que le schéma fournit « a […] aidé pour savoir où l'on attaquait le problème. […] Là, on attaquait sur *identity*, donc on cherchait sur *identity*... ». D'après cette remarque, le schéma fourni (Figure 254) l'a aidé à localiser l'endroit de la modification. D'ailleurs, cette remarque du participant durant la discussion est confirmée par les faits : pendant la modification dont il parle, lui et son partenaire ont trouvé rapidement (1 minutes et 47 secondes) la partie à modifier dans le code. Ainsi, il est possible de se servir de la ressemblance entre schéma et code OP pour faire des modifications du prototype.
Pendant la discussion finale, à la question « est-ce que finalement l'interface graphique ne serait pas ce type de schéma ? » l'un des participants répond : « [le schéma] c'est peut-être une façon de voir le code, mais c'est en aucun cas la grille directrice pour coder. Ça peut nous dire […] juste la partie direction, on peut juste dire la partie *connect* ». Cette remarque suggère que le participant peut avoir compris que la structure est identique, mais que le modèle d'interaction mixte n'est pas une description du logiciel. En effet, la description de l'interaction est traduisible presque directement en ce qui concerne les flèches. Par contre, dans une modélisation d'objet mixte, la forme d'expression d'un élément (comme le langage de liaison) est à l'initiative du concepteur. Ainsi, à partir d'une modélisation, le prototypage n'est pas une évidence : il faut choisir le(s) bon(s) composant(s) OP.

2.3.2.2.2. Limitations concernant le lien entre le modèle d'interaction mixte et l'outil de prototypage

La discussion nous permet aussi d'identifier les difficultés rencontrées. Tout d'abord, le lien entre les flèches et la programmation par évènements est à évaluer plus précisément car l'un des participants dit que le schéma ne l'aide pas pour programmer par évènements.
Ensuite, l'architecture du prototype reprenant les concepts du modèle d'interaction n'est pas complètement encouragé car les deux participants ayant eu le schéma n'ont pas repris le niveau d'abstraction correspondant au langage de liaison. Nous souhaitons à l'avenir les encourager à proposer une solution modulaire et réutilisable pour voir quelle solution de ce type ils proposeraient.
Ensuite, le lien entre le schéma et le prototype n'est pas non plus complètement confirmé et nécessite davantage d'évaluation. En effet, nous ne sommes pas sûrs que ce soit la forme du schéma qui les ait aidé à localiser la modification, car l'un des participants dit « il m'a juste permis de voir

qu'on remplaçait ça par *identity*, du coup j'ai cherche *identity* là-dedans ». Cette remarque suggère que ce n'est peut-être pas le schéma mais le vocabulaire utilisé qui l'a aidé.
Les participants font également des remarques sur la description schématique de l'interaction en tant que structure du logiciel. Fortement influencés par leur expérience en génie logiciel, les participants émettent des réserves sur la forme des schémas en tant que structure du logiciel. Un participant dit que le schéma paraît compliqué « parce que ça mélange plein de concepts différents ». De même un autre participant dit « moi, le formalisme, j'ai eu beaucoup de mal à comprendre, comme je suis UML à fond. J'ai vraiment eu beaucoup de mal à comprendre qu'est-ce qui faisait quoi » et un troisième confirme « j'ai trouvé le schéma difficile aussi. […] Il y a des objets qui pointent vers des variables, et des évènements qui pointent vers des objets ». D'autres remarques vont dans ce sens : « Un seul schéma, c'est trop fouillis, ça mélange pleins de choses séparées d'habitude », ou encore « ça mélange statique et dynamique, objets, actions et variables ». L'un d'entre eux dit qu'il aurait voulu un diagramme d'état, d'évènements, d'architecture, d'équipements. Un participant dit même : « le texte, ce n'est pas des spécifications, c'est juste une observation du fonctionnement de l'objet, alors que le schéma ressemble plus à des spécifications ». Nous comprenons ces remarques dans le contexte du génie logiciel, et nous sommes d'accord que notre description n'est pas adaptée à la conception logicielle, ni à la taille d'application industrielle, car elle peut être ambiguë. L'intérêt de notre description réside dans l'identification de niveaux d'abstraction que nous pensons pertinent pour concevoir l'interaction. Elle n'a pas été pensée pour répondre aux problèmes de conception logicielle, mais pour la conception expérimentale de l'interaction en petite équipe pluridisciplinaire. Un des participants le remarque puisqu'il dit « ce n'est pas assez proche de ce qu'on a l'habitude de voir ».
Sur la forme même de la représentation en tant que structure du logiciel, un participant dit pendant la discussion finale que « ce qui est dérangeant c'est qu'on a les mêmes cercles sur le type d'information, le *device*[53], c'est ça qui est un peu plus troublant finalement ». Une autre remarque va dans ce sens : « on n'arrive pas à imaginer le langage qu'il y a derrière, parce que plusieurs choses sont représentées de la même façon ». Pour savoir s'il est utile de développer un formalisme strict pour l'expression du modèle d'interaction mixte en tant que structure du logiciel, nous attendons d'en savoir plus à partir d'évaluations conduites auprès de tous les types de concepteurs ciblés. En effet, nous présentons l'interaction décrite avec le modèle d'interaction sous une forme, mais cette forme n'est pas contrainte. Notre contribution réside dans l'identification des niveaux d'abstraction : au Chapitre 4, nous avons uniquement défini les concepts et niveaux d'abstraction qui le constituent. Ainsi les remarques des participants confirment que pour utiliser le modèle d'interaction mixte en tant que structure logicielle du prototype, il est nécessaire de travailler sur la forme de la représentation du modèle.

Nous reportons maintenant les points positifs et négatifs que nous avons notés concernant l'outil de prototypage lui-même.

2.3.2.2.3. Points positifs concernant l'outil de prototypage

Les participants novices ont apprécié que la complexité du matériel soit cachée. L'un d'entre eux dit : « J'imagine que […] ça cache l'interface des équipements, alors que c'est toujours quelque chose que j'ai évité d'avoir à mettre, des équipements matériels sur mon code. En fait, quand j'ai eu à le faire, je l'ai filé à quelqu'un d'autre ». Pendant la discussion, à la question « est-ce que cet outil simplifierait ta manière de programmer ? », l'un des participants répond : « si j'avais à utiliser des équipements externes, le fait que ce soit unifié […], oui, évidemment ». La simplicité apparente du code à manipuler a donc été soulignée.

L'un d'entre eux dit que « c'est plus le côté « *je connecte des trucs existants* » qui est intéressant plutôt que le fait qu'il y ait des types d'équipements qui nous cachent la complexité ». En effet, les participants ont apprécié l'utilisation des *connect* de Qt, ce qui était nouveau pour eux. L'un d'entre eux dit que « le fait de pouvoir déclarer les liaisons et puis de dire […] j'ai ça en entrée, et puis je

[53] dispositif

réutilise les composants, et par composition j'arrive à décrire une mini appli, ou du moins un bout d'appli, ça, c'est intéressant ». Pour plusieurs d'entre eux, ce n'était pas écrire du code, car le prototype que nous leur présentions était simple. Un participant dit même « je me demandais si c'était juste les spécifications ou si c'était vraiment quelque chose qui fonctionne » et « là, quelque part, en fait, on n'écrit pas de code. On déclare des relations. C'est ça qui est intéressant parce que, pour moi, tout l'intérêt d'un langage comme ça, c'est que finalement, on veut créer des applications sans rien développer, parce qu'on a déjà les *drivers* pour tous les équipements, et éventuellement on a déjà les fonctions de calculs. Donc on veut juste dire « bah je fais cette application en particulier donc je reprends ça, je reprends ça, hop, je les mets ensemble d'une certaine façon ». Le même sujet ajoute que « ce langage est plus fait pour décrire une architecture que des fonctionnalités, que pour écrire du code ». Un autre participant dit que les modifications demandées, « c'est une ligne à supprimer ou c'est deux petites variables à changer. La modification du code, elle n'a pas l'air complexe… pour ce qui a été demandé ». Par toutes ces remarques, nous voyons que pour ces informaticiens, bien que novices aux outils de bases de OP, l'outil de prototypage leur semble intéressant car il y a peu de code à manipuler, mais des éléments à assembler.

Au-delà de ces remarques encourageantes pour notre outil de prototypage, nous présentons des limitations identifiées pendant cette première expérience en laboratoire.

2.3.2.2.4. Limitations concernant l'outil de prototypage

Tout d'abord, ce que des participants avaient apprécié dans l'encapsulation de la complexité, un autre le nuance en remarquant que « je ne sais pas à quel point c'est unifié et à quel point on peut unifier » entre les différentes ressources matérielles. Il est vrai que nous ne cherchons pas à unifier la manipulation logicielle des ressources matérielles comme le matériel Interface-Z, Phidgets ou les caméras via l'ARToolKit, mais plus à les encapsuler. Le participant, appuyé par un autre, ajoute que « le fait qu'il y ait des types d'équipements qui nous cachent la complexité, [….] il existe autant qu'on veut, en plus il y a des standard qui sortent, donc pour moi, ça n'a pas autant d'intérêt que ça. » Ils citent ensuite DirectX pour les entrées/sorties, ou encore UPnP. Étendre notre outil pour qu'il autorise par exemple les protocoles UPnP pourrait en effet constituer une perspective de ce travail. Notre contribution, même si elle peut bénéficier de cette remarque, ne réside pourtant pas dans l'unification des ressources matérielles, mais dans la conception et le prototypage des interfaces.

Ensuite, nous avons remarqué que les participants ont eu des difficultés à faire la première modification (Figure 253) : changer la réaction visuelle avec la LED pour une réaction sonore avec un haut-parleur. Ils ont compris ce qui leur était demandé, mais ils ont cherché un composant OP pour le haut-parleur. Or OP ne fournit pas de composants pour les dispositifs standards. Nous remarquons d'après leurs difficultés à faire la modification que nous devons améliorer la cohérence avec le modèle, entre dispositifs non conventionnels (comme la LED) ou conventionnels (comme le haut-parleur). En effet, ces dispositifs sont décrits de la même façon en conception, mais les premiers sont gérés par OP alors que les seconds sont juste utilisés, directement via le langage de liaison. Nous projetons de proposer des composants OP pour utiliser les dispositifs standards.

Ensuite, nous avons remarqué que plusieurs participants pensaient que *BeepOutputLanguage* impliquant forcément un retour sonore, à cause de son nom. Nous pensons donc améliorer la nomenclature des composants. Par exemple, renommer *BeepOutputLanguage* en *ImpulseOutputLanguage*.

Une autre difficulté résidait dans la documentation fournie. Nous avons aussi observé que les participants n'ont souvent pas trouvé les informations qu'ils cherchaient dans la documentation. Ils l'ont d'ailleurs critiqué pendant la discussion finale : « la documentation, elle n'est pas bien fournie et elle n'apporte pas grand-chose ». Nous l'avons modifiée avant l'expérience suivante.

Un participant a émis aussi des réserves sur le fait qu'il ne voyait pas directement ce qu'il est possible de connecter. Il dit : « Tu peux définir un lien entre n'importe quoi et n'importe quoi. Alors que, en fait, on voudrait être contraint, parce qu'il y a des contraintes […] qui sont qu'un *InputDevice*, tu ne peux pas le connecter avec n'importe quoi […]. Du coup, il faut découvrir ». Ce type de contraintes doit être résolu en regardant la documentation. Nous envisageons aussi de

rendre ces contraintes perceptibles avec le développement d'une interface graphique qui les représenterait graphiquement.

En plus de ces remarques sur l'outil OP, nous avons également retenu les remarques des participants relatives au protocole de l'expérience. Nous les présentons maintenant.

2.3.2.2.5. Remarques sur le protocole

De façon générale, nous avons remarqué des problèmes de compréhension des consignes. En effet, en définissant le protocole, nous avions décidé de ne rien expliquer pour pouvoir identifier sans biais les difficultés rencontrées.
Les participants de cette expérience ont cru que le code fourni à l'exercice 2 était celui qui correspondait au premier exercice. Nous avons donc décidé pour l'expérience suivante d'assouplir cette décision pour certaines consignes. Nous avons précisé que les deux objets, décrits au premier exercice ou prototypé au second exercice, ne sont pas les mêmes.
La compréhension de la consigne de la seconde modification leur a pris beaucoup de temps. Des remarques vont dans ce sens, comme « on a buté sur la formulation du problème » en parlant de la modification demandée (Figure 254). L'un des participants propose d'écrire la modification dans la Figure 254 sans barrer le terme *identity* mais en écrivant « *identity* + <modification> ». Nous pensons que le but n'est pas d'aider les participants artificiellement pour biaiser les résultats. Nous avons décidé pour la seconde expérience d'expliquer la modification à faire aux participants en leur faisant une démonstration et en répondant à toutes leurs questions, plutôt qu'en leur fournissant des schémas (Figure 253 et Figure 254). Nous pensons que nous obtiendrons déjà des retours sur le lien entre le schéma et le prototype avec les premiers exercices.
Les sujets ont fait également remarquer que les descriptions de la Figure 250 ne donnaient pas la même information. Pendant la discussion après les deux premiers exercices, la première réaction d'un participant qui a eu une description texte en voyant le schéma des autres a été « c'est pour ça ! forcement ! ». Il a expliqué sa réaction en disant qu'il venait de comprendre pourquoi les autres parlaient d'un seuil, alors que lui ne savait pas s'il y en avait un ou deux. Nous avons corrigé cet aspect pour la seconde expérience en modifiant le texte (Figure 257).
Les participants font aussi des remarques sur l'apprentissage : « Peut-être que la période d'apprentissage peut être plus ou moins longue, mais après il faut voir aussi si le temps d'apprentissage n'apporte pas un bénéfice à la suite dans la réutilisation du logiciel ». Il insiste en disant que « tu as peut-être de la perte de temps au démarrage, mais après tu as peut-être du gain de temps. Enfin ça je sais pas, il faudrait faire plusieurs fois ». Un autre participant va dans ce sens en disant que « l'apprentissage pour nous, parce qu'on est habitué à fonctionner d'une certaine façon, il est compliqué. Mais peut-être qu'une fois qu'on a bien capté le principe, finalement, on peut peut-être utiliser ça plus facilement ». Ces remarques sur l'apprentissage montre que ces expériences en laboratoire sont peu réalistes. Elles nous servent à recueillir les premières impressions, et ne peuvent pas prendre plus d'importance. Pour évaluer plus précisément l'utilisation de l'outil, il nous faudra mettre OP dans les mains de sujets plus longtemps.

Nous venons de présenter la première expérience effectuée, ses résultats et les améliorations que nous avons apportées dans le protocole. Nous présentons maintenant la seconde expérience effectuée, selon ce protocole modifié.

2.3.3. Seconde expérience

La Figure 249 montre une image issue de la vidéo de l'expérience avec des informaticiens connaissant Qt. Sur les deux images, les participants essaient, deux par deux, de faire la modification demandée. À gauche ils sont en train d'essayer leur solution en manipulant l'objet mixte (bleu). À droite, ils cherchent des renseignements dans la documentation. Une expérimentatrice les observe à l'arrière-plan.



Figure 256 : Expérience avec des informaticiens connaissant Qt. À gauche, manipulation du prototype en bleu. À droite, recherche d'information dans la documentation.

Nous présentons d'abord le déroulement et les supports que les participants ont eu pour réaliser les exercices, puis les résultats obtenus dans ces conditions.

2.3.3.1. Déroulement

Pour le premier exercice, nous avons modifié la description textuelle d'après les résultats de l'expérience précédente : la Figure 257 présente, surlignées en gris, les modifications effectuées dans le texte qui ajoute l'information sur le seuil.

> Marie manipule un objet sensible à la lumière. Il contient un capteur de luminosité et est muni d'une diode électroluminescente (LED) à sa surface. Lorsqu'elle approche ~~trop~~ cet objet d'une source lumineuse au-delà d'un certain seuil, la diode clignote une fois. Alors, Marie éloigne l'objet de la lumière au-delà du seuil : la diode clignote une nouvelle fois quand l'objet est suffisamment loin de la source lumineuse.

Figure 257 : Changements effectués dans la description textuelle de l'objet mixte pour la première expérience.

La description selon le modèle d'interaction mixte (Figure 250, à droite) et le code du second exercice (Figure 251) n'ont pas été modifiés. En revanche, nous avons modifié notre façon de présenter les modifications à effectuer d'après les résultats de l'expérience précédente. Comme exposé précédemment, nous avons décidé de leur faire la démonstration concrète de ce que nous voulions obtenir, en manipulant concrètement l'objet à modifier. Nous avons également répondu à leurs questions et nous nous sommes assurés qu'ils avaient compris la modification avant de commencer à la réaliser. Nous avons également modifié le code à modifier, en utilisant *IdentityInputLanguage* à la place de *RampInputLanguage*. En effet, d'après les résultats de l'expérience précédente, nous avons décidé d'utiliser le composant OP le plus simple : même si *RampInputLanguage* peut exprimer la même transformation que *IdentityInputLanguage*, ce dernier est plus évident. Le nouveau code de départ est présenté à la Figure 258.

```
int indexOnBoard = 0;
bool isAnalogue = true;
int resolution = 7;
MIDIDevice turnSensor("TurnSensor", LinkingComponent::IN, indexOnBoard, isAnalogue, resolution);

bool isOpposite = false;
int min = 0;
int max = 127;
IdentityInputLanguage identity("identity", isOpposite, min, max);

QObject::connect(&turnSensor, SIGNAL(updated(int, QTime)),
                 &identity, SLOT(update(int, QTime)));

DigitalProperty angle("angle", QVariant::Int);

QObject::connect(&identity, SIGNAL(updated(QVariant, QTime)),
                 &angle, SLOT(updateProperty(QVariant)));

DigitalProperty level("level", QVariant::Int);

char* soundFile = "";
BeepOutputLanguage beep("beep", soundFile);

QObject::connect(&level, SIGNAL(PropertyUpdated(QVariant)),
                 &beep, SLOT(update(void)));

isAnalogue = false;
PhidgetInterfaceKitDevice led("led", LinkingComponent::OUT, indexOnBoard, isAnalogue);

QObject::connect(&beep, SIGNAL(updated(bool)),
                 &led, SLOT(UpdateOutput(bool)));
```

Figure 258 : Code OP à modifier lors de la seconde expérience.

Pour cette expérience, nous avons également intégré les améliorations demandées par les participants concernant la documentation.

Nous présentons maintenant les résultats de cette seconde expérience avec des informaticiens connaissant Qt.

2.3.3.2. Résultats

Nous avons noté à partir de l'enregistrement vidéo les points positifs et les limitations identifiés lors de cette première expérience.

2.3.3.2.1. Points positifs concernant le lien entre le modèle d'interaction mixte et l'outil de prototypage

Tout d'abord, comme dans l'expérience précédente, un participant a proposé un pseudo-code pour le premier exercice contenant une couche qui abstrait du niveau dispositif (classe *Capteur* et classe *LED*). Il commente son code en disant que cette solution lui paraît plus générique. Ce même participant dit qu'il « préfère le schéma, parce que ça donne une vision plus globale du problème et ça permet de raccrocher ça à un problème similaire qu'on a déjà vu avant ». Un autre participant l'interpelle : « maintenant je te donne le même schéma, mais j'enlève toutes les écritures, est-ce que tu serais capable de remettre des trucs un peu génériques dans les cases ? », ce à quoi il répond : « Oui je pense, oui ». Ceci semble confirmer l'architecture de l'outil OP reprenant les concepts du modèle d'interaction mixte.
De plus, ils confirment le besoin de schéma pour programmer et l'un d'entre eux dit : « je pense que quand tu as un texte, tu es obligé de te faire une représentation mentale qui est, un petit peu, finalement, un schéma ».

Pour cette expérience et pour expliquer le code de la Figure 251, nous avons explicitement proposé aux participants de faire un schéma. Trois d'entre eux ont effectivement choisi de faire un schéma. Leurs représentations sont présentées à la Figure 259, à la Figure 260, et à la Figure 261. Nous constatons que dans tous ces schémas, nous retrouvons les niveaux d'abstraction manipulés dans le code et donc dans le modèle d'interaction mixte. De plus, le participant qui a fait le schéma de la Figure 260 dit « ça me paraît naturel de le dessiner comme ça ». D'autre remarques vont dans ce sens, comme « pour moi, il y avait un *slider* qui positionnait une variable [propriété numérique] *level*, qui pour moi était le centre qui stockait l'information, et on prévenait l'utilisateur que ça avait changé ». À la question « vous avez eu un diagramme avant, est-ce que vous avez vu le lien avec le code ? », un des deux participant concerné répond que « oui je l'ai clairement vu. […] J'hésitais presque à faire le même schéma ». L'expérimentatrice lui demande s'il serait « prêt à [se] lancer à faire exactement le même schéma qu'[il avait] avant », ce à quoi il répond : « ah oui, pas de souci ». Ces remarques suggèrent que la correspondance entre les concepts manipulés dans le code et les concepts représentés dans le modèle d'interaction mixte est facile à faire. De plus, comme pour l'expérience précédente, nous avons remarqué que les participants qui ont eu la description sous forme de modèle (Figure 250, à droite) ont terminé l'exercice d'explication de code en premier.

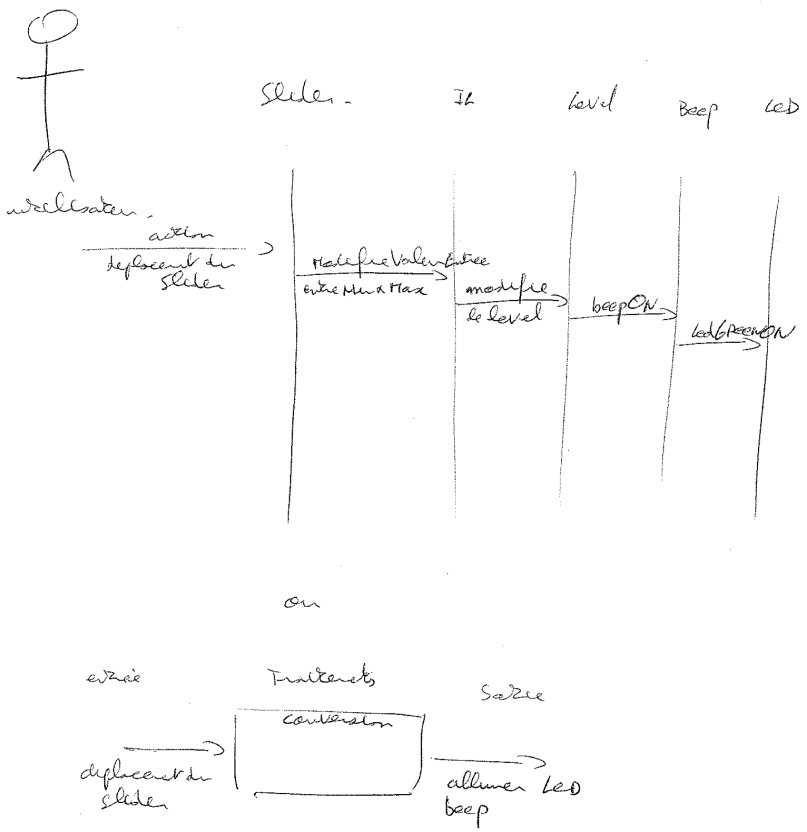


Figure 259 : Schémas réalisés par un participant de la deuxième expérience pour expliquer le code de la Figure 251.

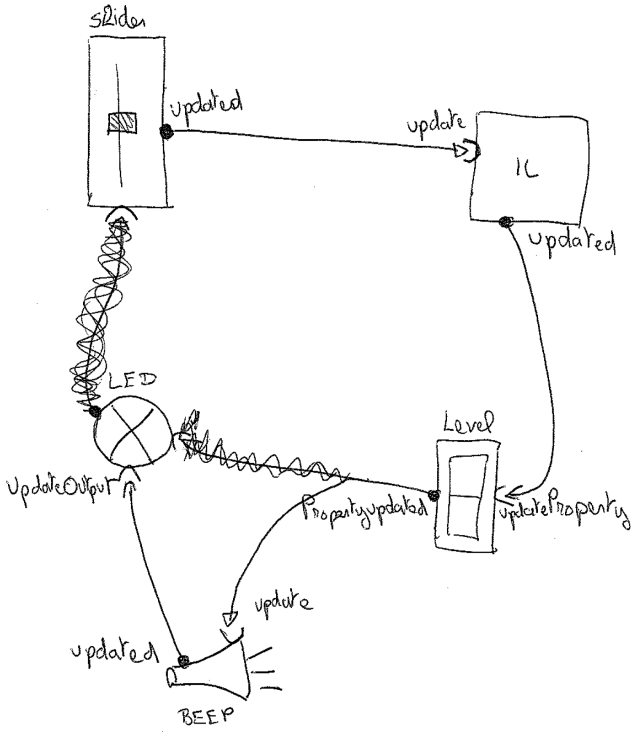


Figure 260 : Schéma réalisé par un participant de la deuxième expérience pour expliquer le code de la Figure 251.

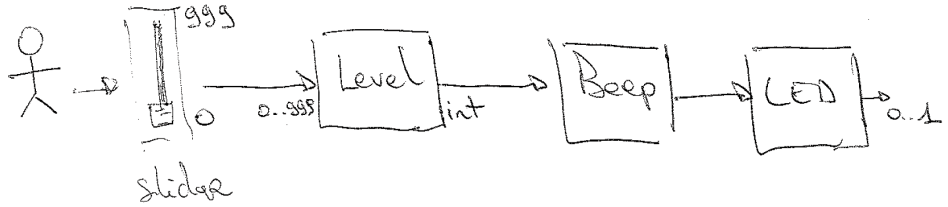


Figure 261 : Schéma réalisé par un participant de la deuxième expérience pour expliquer le code de la Figure 251.

2.3.3.2.2. Limitations concernant le lien entre le modèle d'interaction mixte et l'outil de prototypage

Tout d'abord, certaines remarques faites dans l'expérience précédente sont confirmées. Par exemple les pseudo-codes proposés sont très peu modulaires. Les participants expliquent cela par la simplicité du problème qu'on leur soumet : ils ne voient pas l'intérêt de chercher la généricité pour un problème si petit. Nous souhaitons à l'avenir les encourager à proposer une solution modulaire et réutilisable pour voir quelle solution de ce type ils proposeraient. Pour cela, nous envisageons de les projeter davantage dans une situation de prototypage rapide.
Comme pour l'expérience précédente, les participants font également des remarques sur la description schématique de l'interaction en tant que structure du logiciel et émettent des réserves sur la forme des schémas. Par exemple l'un d'entre eux dit « On dirait qu'il y a derrière un formalisme, mais il n'est pas explicite ». Il nous semble donc nécessaire de rechercher une forme de présentation du modèle d'interaction mixte qui montre clairement que l'intérêt de notre description réside dans l'identification de niveaux d'abstraction que nous pensons pertinents pour concevoir l'interaction, et ne réside pas dans la réponse aux problèmes de conception logicielle.

Un participant dit que pour programmer, sa représentation mentale est « séquentielle […]. Tu vois un déroulement d'évènements temporellement ». Pour écrire son pseudo-code, il a schématisé le fonctionnement de l'objet sensible à la lumière sous la forme présentée à la Figure 262. La

description de l'interaction de la Figure 262 ne lui a donc pas paru séquentielle. Les signaux à la Figure 262 représentent pourtant l'information qui transite dans l'objet mixte :

- en haut, la transformation effectuée par le composant de langage en entrée (seuil),
- au centre, l'état de la propriété numérique « est exposé » (Figure 250, à droite),
- en bas, la transformation effectuée par le composant de langage en sortie (impulsion).

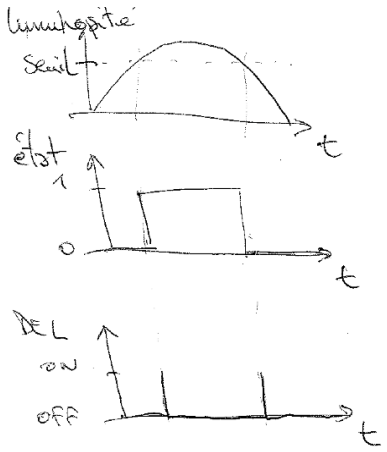


Figure 262 : Schéma réalisé par un participant de la deuxième expérience avant d'écrire le pseudo-code à partir du texte de la Figure 257.

Nous reportons maintenant les points positifs et négatifs que nous avons notés concernant l'outil de prototypage lui-même.

2.3.3.2.3. Points positifs concernant l'outil de prototypage

Nous soulignons tout d'abord que pour cette expérience, les participants ont été d'accord pour dire que la documentation (modifiée depuis l'expérience précédente) était bien faite et qu'ils y ont trouvé ce qu'ils cherchaient.

Nous avons relevé les observations et les remarques sur la difficulté de la modification. Un participant dit qu'« on sent bien que le bout de code, il n'y a que deux lignes à faire… une ligne ». Lui et son partenaire ont d'ailleurs mis 1 minute et 30 secondes pour localiser le lieu de la modification. Un participant de l'autre binôme dit « je suis remonté dans la hiérarchie [dans la documentation]. Et là […], tu avais la liste de tous les trucs que tu pouvais interchanger avec le *InputLanguage*, et là, on a trouvé le *Ramp*, on l'a mis à la place et voilà ». Bien que les participants n'aient pas réussi à faire la modification, nous avons remarqué que dès que l'expérimentatrice les a aidés, ils ont mis 2 minutes et 40 secondes pour trouver, faire la modification, compiler et constater sur l'objet physique que leur solution fonctionne. L'aide fournie aux participants était la remarque suivante : « Vous avez essayé de changer les valeurs des paramètres, mais vous pourriez faire autre chose. Par exemple, changer le type des objets que vous êtes en train de manipuler ». Une fois la solution donnée à l'autre binôme, un participant s'exclame « ah comme c'est beau! » et dit « maintenant qu'elle nous a expliqué la solution, non [je ne trouve pas ça difficile] ».
Ces remarques et ces observations suggèrent que la solution leur paraît facile. Nous présentons les limitations de cet aspect dans la section suivante. Pour vérifier que ce n'est pas le protocole qui rend la modification trop difficile, nous voulons former à OP les participants aux prochaines expériences.

Comme pour l'expérience précédente, les participants apprécient l'abstraction du dispositif. Un participant dit « il y a une grosse couche d'abstraction derrière : un petit bouton qui clignote sur l'écran, ça serait programmé pareil ». Un autre dit : « La programmation du capteur […], on ne la voit pas, […] ce qui est bien : on avait un objet qui envoyait un truc, et ça c'était très bien ».
L'unification des différentes ressources matérielles a fait débat, comme pour l'expérience avec les novices. Lorsqu'un participant dit que « tu peux avoir un standard mais qui est ouvert malgré tout », un autre ajoute que « c'est plutôt ce qu'ils ont fait […], c'est-à-dire remonter dans la hiérarchie des

objets du même type pour le changer. Et le spécialiser autrement ». Le niveau d'abstraction choisi pour les éléments de l'outil OP semble donc avoir été apprécié.

Nous avons également relevé leurs observations relatives à Qt, puisque la particularité de ces participants était qu'ils connaissaient Qt, alors que les précédents étaient novices. Un participant fait remarquer qu'il apprécie l'utilisation des signaux et des slots de Qt et les compare à Gtk : « je trouve que c'est bien, tu les vois dans la classe, et d'ailleurs dans la documentation, ils apparaissent super bien : tu vois *signaux*, *slots*. Tu vois ce que tu peux connecter. Alors que dans Gtk, c'est pas si clair que ça, ce que tu peux connecter ».
L'expérimentatrice demande à l'un d'entre eux : « Vu ton expérience, si on te demandait maintenant de développer une interface physique, qu'est ce que tu penserais de l'utilisation de cet outil ? » Il répond que « ça ne change pas tellement de développer une interface qui soit graphique ou physique, parce que finalement c'est juste des blocs qu'on va assembler. [...] Pour moi, ça aurait été un *slider* graphique à l'écran et puis un bouton qui change d'image à l'écran, ça aurait été similaire ». Ceci confirme notre choix d'extension de Qt.

Au-delà de ces remarques encourageantes pour notre outil de prototypage, nous présentons des limitations identifiées pendant cette première expérience en laboratoire.

2.3.3.2.4. Limitations concernant l'outil de prototypage

Certaines observations et remarques de cette expérience viennent confirmer les difficultés identifiées lors de la précédente. Tout d'abord, les participants n'ont pas compris que *BeepOutputLanguage* pouvait être général et pas seulement sonore. En effet pour cette seconde expérience, nous n'avions pas encore changé son nom.
De plus les participants de cette expérience confirment le désir d'avoir une interface graphique. Un participant dit : « Je pense qu'il faut le faire une bonne fois pour toute, d'écrire chaque composant et après avoir une interface graphique pour faire avec la souris ». Un autre participant voudrait « quelque chose de graphique, où on voit bien ». À cette idée s'ajoute une nouvelle : « si j'avais à coder quelque chose rapidement, la question que je me poserais ce serait plutôt langage interprété versus langage dynamique. [...] Moi j'aurais fait Ruby/Qt plutôt que C++/Qt ». Ceci est une autre perspective d'extension d'OP.

Ils ont aussi trouvé l'exercice « pas facile » et « délicat ». Nous rappelons que l'expérimentatrice a aidé un des binômes pour l'exercice de modification, en disant : « vous avez essayé de changer les valeurs des paramètres, mais vous pourriez faire autre chose. Par exemple, changer le type des objets que vous êtes en train de manipuler ». À l'issue de cet exercice, un participant dit « sans aide, on n'y serait pas arrivé ». Même après les avoir aidé, l'autre participant ajoute : « si tu n'as pas de documentation, point de salut ».
Néanmoins, la difficulté de l'exercice est à nuancer, car aucune formation n'était donnée au préalable aux participants. Ils le soulignent en disant que « sans connaissance a priori, c'est délicat, il y a toujours un nombre infini de façon de faire la même chose ». Il ajoute qu'« il faut l'apprendre. Il n'y a pas de secret, tu ne peux pas demander à ce que, sans pré-connaissance, on puisse se débrouiller ».

Enfin, pendant la discussion après l'exercice d'explication de code, les participants font des remarques à propos de Qt :
- « Je trouve ça infâme comme code [en parlant du code de la Figure 251]. C'est illisible.
- Ah bah ! C'est du Qt !
- Oui, j'ai fait du Qt et j'avais eu la même réflexion »
Pour mieux comprendre ces remarques, l'expérimentatrice demande :
- « Qu'est-ce que tu n'aimais pas dans Qt ?
- Ce n'est pas explicite. Et là encore, je pense que c'est juste un petit bout, il y en a beaucoup d'autre ailleurs du code. Quand il y a 4 lignes [en montrant le code OP], ça va, mais quand on

commence à avoir 50 lignes et qu'on veut modifier une chose bien précise ou qu'on veut modifier un peu le comportement global, on doit tout reprendre. Ce n'est pas du tout intuitif. »
Nous soulignons que nous n'avons encore jamais rencontré de situations demandant de prototyper un objet mixte avec un nombre important de lignes de code, et que nous ne pensons pas confronter les utilisateurs de l'outil OP à un code trop important.

En plus de ces remarques sur l'outil OP, nous avons également retenu les remarques des participants relatives au protocole de l'expérience.

2.3.3.2.5. Remarques sur le protocole

De façon générale, les participants de cette expérience ont mieux compris les consignes. Les participants de cette expérience n'ont pas cru que le code fourni pour le deuxième exercice était celui qui correspondait au premier exercice. Ils n'ont donc pas eu ce biais.
Les participants soulèvent également le problème de l'apprentissage. La confirmation de ces remarques sur l'apprentissage montre que notre protocole n'est pas assez réaliste de ce point de vue. De plus, elles soulignent le besoin d'étudier comment les sujets utilisent une approche analogique et réutilisent des éléments OP lors de la conception.
Nous retenons également que les sujets font des remarques montrant qu'ils ne projettent pas dans une situation de réalisation d'ébauches. Il nous paraît donc nécessaire d'améliorer le protocole en proposant un scénario les mettant dans cette situation.
Pendant l'exercice de modification, nous avons observé qu'au maximum, les participants ont pu réaliser quatre itérations (modification, compilation, exécution du code et test du prototype physique) pour essayer quatre solutions différentes pendant les 10 minutes. Un sujet commente : « c'est un peu court pour juger en 10 minutes » et il est approuvé par un autre participant. Au contraire, nous pensons que 10 minutes pour faire une modification est réaliste, en nous basant sur notre expérience et les expériences en situation réelle qui ont été réalisées précédemment. Pour remédier à ce problème, il est nécessaire lors des prochaines évaluations de former les participants à OP avant qu'ils le manipulent.

Nous venons de présenter la seconde expérience effectuée et ses résultats. Les points positifs identifiés pendant cette expérience et la précédente nous ont permis de confirmer les choix faits pour le développement d'OP. Les limitations nous permettent d'identifier les aspects de l'outil OP qui nécessitent d'être retravaillé, étendus, ou évalués une nouvelle fois. Ces évaluations nous permettent donc de nourrir les perspectives de nos travaux.

3. Synthèse

Nous avons présenté dans ce chapitre nos actions et résultats pour évaluer les apports et les limites de notre outil OP. La Figure 263 résume les évaluations effectuées. Nous nous sommes appuyés sur un état de l'art des cadres d'évaluation d'outils pour l'Interaction Homme-Machine pour mieux cerner les différentes formes d'évaluation possibles de l'outil OP. Nous avons d'abord appliqué les cadres d'évaluation de [Olsen, 2007] et [Greenberg, Buxton, 2007] pour justifier l'importance du problème traité par OP et notre démarche pour concevoir l'outil. Nous avons ensuite mené des expérimentations, à la fois en situation réelle et en laboratoire.

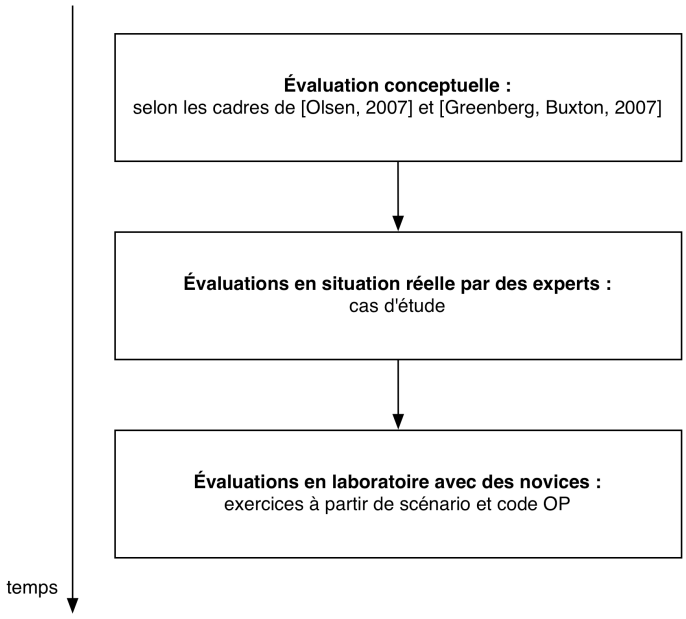


Figure 263 : Évaluations de l'outil OP.

Nous envisageons de réaliser des évaluations de l'outil de prototypage OP avec des utilisateurs novices informaticiens pendant plusieurs séances, afin d'observer l'apprentissage de l'outil. Nous envisageons également de réaliser les mêmes expériences avec les Phidgets comme référence de comparaison.

Chapitre 9 : Conclusion et Perspectives

Les travaux présentés concernent la conception et le prototypage des interfaces mixtes, qui allient les avantages du monde physique et du monde de l'ordinateur. L'objectif de notre travail est de fournir aux concepteurs d'interfaces mixtes un cadre uniforme pour l'exploration systématique de l'espace de conception. Pour cela, nous avons étudié et défini des outils pour les deux facettes de la conception que définissent les activités conceptuelles et pratiques de conception.

Pour conclure ce manuscrit, nous rappelons nos contributions pour la conception des systèmes de réalité mixte. En accord avec nos objectifs initiaux, ces contributions concernent les aspects conceptuels et pratiques de la conception. Nous identifions ensuite les limites de ce travail, puis ses perspectives à moyen terme, et enfin deux ouvertures précises à long terme que nous donnons à nos travaux.

1. Résumé des contributions

1.1. Contribution pour la conception : le modèle d'interaction mixte

Pour mieux comprendre, cerner et contribuer à l'espace de conception des interfaces mixtes, nous avons introduit au Chapitre 4 un modèle d'interaction, appelé modèle d'interaction mixte. Ce modèle permet :

- de définir les objets mixtes, à la fois physiques et numériques, qui entrent en jeu dans l'interaction entre l'utilisateur et le système ; nous avons identifié des niveaux d'abstraction pour caractériser un objet mixte ;
- de définir l'interaction entre l'utilisateur et le système via des objets mixtes.

Le point de vue du modèle d'interaction mixte centré sur les objets mixtes nous permet de jeter un regard nouveau sur les interfaces mixtes mais aussi unificateur des travaux existants. Pour cela, nous avons montré les pouvoirs descriptif, comparatif et génératif du notre modèle :

- Tout d'abord, son pouvoir descriptif nous a permis de décrire un large éventail d'interfaces mixtes. De plus, même si le modèle ne permet pas toujours de décrire aussi bien certaines interfaces, nous n'avons toujours pas identifié d'interface mixte qui ne pouvait pas être décrit avec le modèle d'interaction mixte.
- Ensuite, nous avons défini des caractéristiques intrinsèques et extrinsèques aux objets mixtes, qui capitalisent, étendent ou identifient de nouvelles dimensions de conception (Tableau 32). Les caractéristiques sont soit indirectement exprimées soit explicites dans le modèle d'interaction mixte (Tableau 32). En nous appuyant sur ces caractéristiques, nous avons étudié le pouvoir comparatif du modèle en considérant et en classant des systèmes de réalité mixte similaires, montrant ainsi le fort pouvoir de classification de notre modèle.

Caractéristique	**Indirectement exprimée**	**Explicite**
Nouvelle	Propriétés numériques à rebond	Propriétés numériques acquises et perceptibles Ports d'entrée numérique
Étendue de l'existant	Degré d'intégration Caractérisation spatiale et temporelle des propriétés physiques Caractérisation spatiale et temporelle des objets entre eux	Propriétés physiques captées et générées Ports de sortie physique Langages d'interaction Métaphore(s) de nom
Capitalisée de l'existant	Couplage des modalités de liaison Affordance des propriétés physiques Propriétés physiques à rebond	Dispositifs de liaison Langages de liaison Liaisons multimodales Rôle Ports d'entrée physique Ports de sortie numérique Métaphore de verbe

Tableau 32 : Résumé des caractéristiques manipulées dans le modèle d'interaction mixte.

- Son pouvoir génératif nous a permis d'explorer de façon systématique l'espace de conception. Pour cela, nous avons défini des solutions de conception en considérant les valeurs possibles des caractéristiques identifiées précédemment. Nous avons évalué le pouvoir génératif du modèle d'interaction mixte en l'utilisant lors d'expériences en situation réelle de conception.

1.2. Contribution pour le prototypage : la boîte à outils OP

Fidèle à notre démarche qui considère la conception comme une activité entrelacée de phases conceptuelles et pratiques, nous avons présenté une boîte à outils, appelée OP, pour matérialiser facilement les solutions de conception identifiées et décrites avec le modèle d'interaction mixte. Les points saillants de cette contribution sont les suivants :

- OP fournit des éléments réutilisables reprenant les concepts manipulés par le modèle d'interaction mixte. L'outil permet ainsi de disposer d'objets mixtes ou de composants d'objets mixtes, pouvant prendre part à l'interaction avec une application en tant qu'outil ou objet de la tâche.
- OP permet d'augmenter le niveau de détails des premiers prototypes en mettant facilement à disposition des concepteurs des technologies avancées. Pour cela, nous n'avons pas cherché à résoudre un problème technologique local, mais nous avons intégré des technologies rencontrées dans les systèmes de réalité mixte, comme des dispositifs d'électroniques ou des éléments de vision par ordinateur pour la réalité augmentée. Ainsi les premiers prototypes peuvent utiliser ces technologies et être plus proches de la solution finale.
- OP intègre des boîtes à outils existantes comme les Phidgets et l'ARToolKit, fidèle à notre démarche d'unification des propositions du domaine. Cet aspect montre aussi que la boîte à outils peut être étendue à d'autres technologies.
- OP facilite une évolution rapide du prototype en facilitant les modifications et les micro-itérations. Pour cela, nous avons construit des composants élémentaires paramétrables. De plus, nous nous sommes basés sur Qt (http://www.qtsoftware.com) et nous avons exploité son mécanisme de connexion entre éléments. Enfin, nous avons mis à la disposition du développeur une interface graphique pour la mise au point ou la simulation des prototypes.
- OP permet l'exploration de l'interaction et de l'apparence physique des objets mixtes. Pour cela, OP n'impose aucune contrainte sur les matériaux à utiliser, et surtout permet aux concepteurs de se concentrer sur cet aspect matériel de la conception.

Nous présentons dans les sections suivantes les limites et extensions à court terme de ces travaux, avant de présenter nos perspectives à moyen et long terme.

2. Limites identifiées et extensions à court terme

Les évaluations de nos travaux nous ont permis d'en identifier les limites.

Pour le modèle d'interaction, nous avons souligné au Chapitre 4 que certaines caractéristiques existantes ne sont pas encore capitalisées/étudiées au sein du modèle. Par exemple, pour intégrer le mode d'interaction (passif, actif) [Renevier, 2004] de l'utilisateur avec les objets mixtes, il convient d'étudier le niveau d'abstraction concerné du modèle. Nous avons également identifié un frein à l'utilisation du modèle : sa compréhensibilité par les utilisateurs novices. Nous travaillons à une présentation adaptée du modèle pour son utilisation en séances de conception. De plus, nous

envisageons de mener plus d'évaluations de la compréhensibilité du modèle afin de mieux cerner quels aspects du modèle sont difficiles à comprendre.

Pour l'outil de prototypage OP, nous en avons identifié des limites au Chapitre 8.

- Tout d'abord, nous avons souligné pendant les études en situation réelle que la bibliothèque de ses éléments pouvait encore être étendue. Cette limitation est en court de résolution puisque nous travaillons actuellement à l'intégration d'Arduino, mais aussi au développement d'autres types de composition des modalités de liaison comme la redondance ou l'équivalence.
- Ensuite, la transition fluide entre conception et prototypage (contribution principale de l'outil OP) reste difficile à évaluer. Nous avons relevé le défi en effectuant un premier pas pour l'évaluer (Chapitre 8). L'expérimentation que nous avons menée a été sans une formation préalable des utilisateurs novices à OP. Nous en avons vu les limitations : les utilisateurs ont touché du doigt le potentiel de l'outil OP, mais ont rencontré des difficultés en pratique. Nous mettons donc actuellement en place une deuxième série d'évaluations en formant les utilisateurs avant de les mettre en situation de prototypage rapide.
- Une autre limitation d'OP est qu'il ne prend pas en compte la totalité de l'activité de conception puisque la phase d'analyse du prototype n'est pas intégrée. Nous sommes donc en cours de développement d'un outil annexe pour instrumenter l'évaluation via la génération optionnelle de traces. Nous envisageons un outil de visualisation de ces traces intégrées à la vidéo des expérimentations. Ainsi nous pourrions davantage accompagner les micro-itérations effectuées pendant la conception.
- Enfin, une autre limite de notre outil de prototypage est le revers de son intérêt : il permet de prototyper des objets mixtes, mais pas l'interaction complète. Pour pallier à cela, l'insertion de la boîte à outils OP dans un outil de prototypage de l'interaction comme OpenInterface (http://www.oi-project.org/) est en cours de réalisation.

3. Perspectives à moyen terme

3.1. Pour le modèle d'interaction mixte

Pour enrichir le modèle d'interaction mixte, nous envisageons d'étudier les principes et critères d'ergonomie [Bach, Scapin, 2005] au regard des concepts du modèle. L'objectif serait alors de permettre une évaluation prédictive de la solution de conception modélisée. Ainsi le modèle permettrait non seulement de générer de nombreuses solutions de conception, mais aussi de pouvoir choisir celles qui respectent certains critères ergonomiques.

Nous visons une refonte de la présentation du modèle d'interaction mixte, afin qu'il puisse être facilement manipulé. En effet, les difficultés (notamment de compréhension) rencontrées par les concepteurs novices sont un frein à son utilisation plus large, et donc à son enrichissement. Nous envisageons de travailler la présentation du modèle en nous inspirant de techniques de conception d'interaction utilisées en design. Plusieurs pistes sont à explorer, dont celle du jeu notamment : l'écriture collective de scénario à partir de cartes à jouer conçues spécialement à l'image des concepts du modèle d'interaction et permettant de choisir différentes valeurs de caractéristiques. La Figure 264 présente une piste d'adaptation à un tel jeu de cartes : à gauche le dos des cartes, à droite différentes cartes possibles correspondantes.

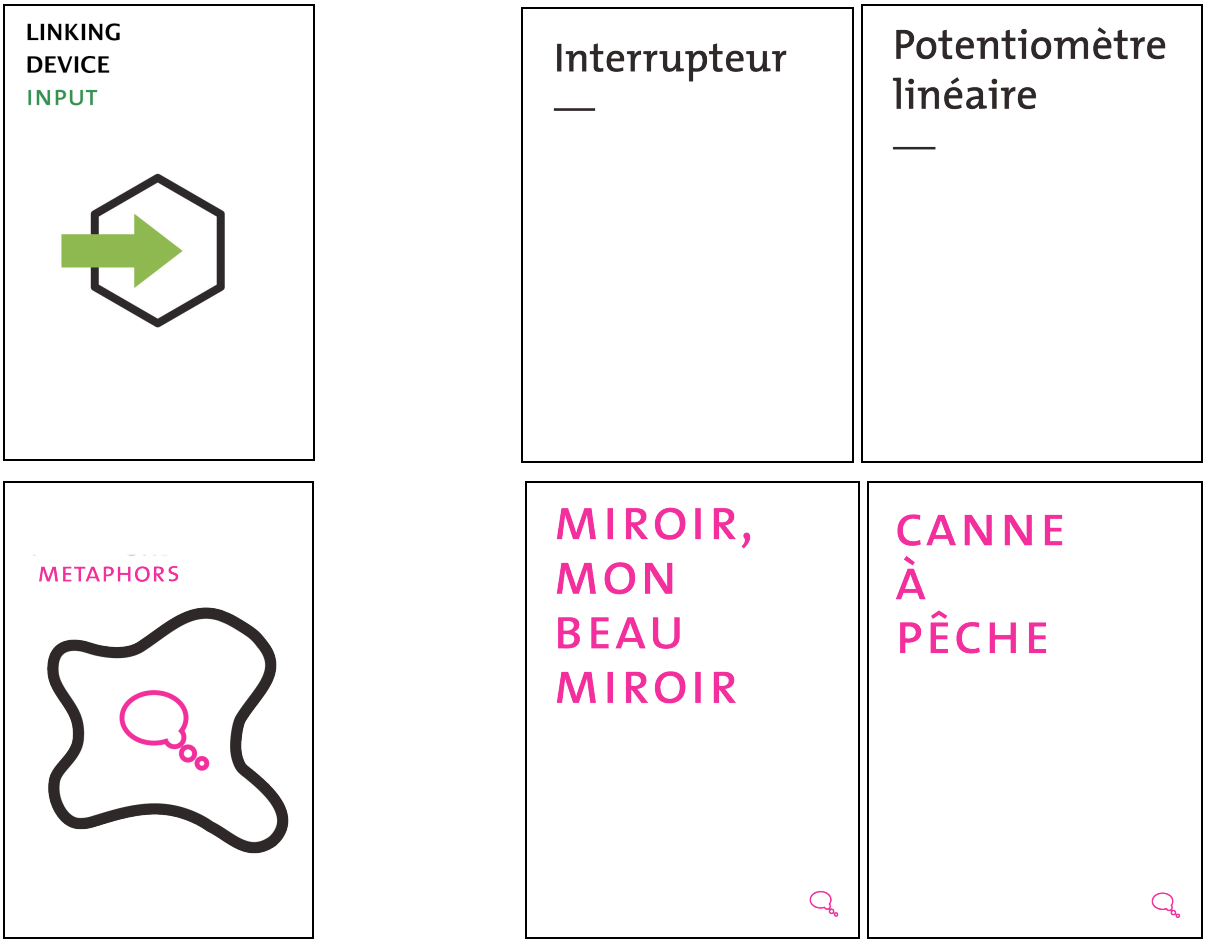


Figure 264 : Jeux de cartes à utiliser pour la conception (d'après Alexandre Elmir, http://www.davectors.com/)

De plus, pour montrer l'intérêt d'utiliser un tel outil pendant la conception, nous envisageons de mener des évaluations avec deux groupes de concepteurs, l'un avec et l'autre sans le modèle, et de leur demander de concevoir de nombreuses solutions.

3.2. Pour l'outil de prototypage OP

Nous envisageons à moyen terme d'étendre un constructeur graphique d'interfaces avec les éléments de OP. Pour cela, plusieurs solutions doivent être considérées, dont par exemple l'outil Qt Designer (http://doc.trolltech.com/4.5/designer-creating-custom-widgets-extensions.html). L'outil graphique résultant permettrait alors de :

- Concevoir un objet mixte rapidement avec une interface qui permet de glisser-déposer des composants à partir d'une palette des éléments d'OP et de les connecter avec la souris.
- Mettre au point précisément les éléments du prototype d'objet mixte en modifiant les propriétés de ces éléments.
- Permettre l'extension d'interfaces graphiques avec des objets mixtes et ainsi conduire progressivement les concepteurs d'interfaces graphiques vers la conception d'interfaces mixtes.

La Figure 265 présente la maquette d'une interface graphique (basée sur Qt Designer) pour le prototypage d'objets mixtes.

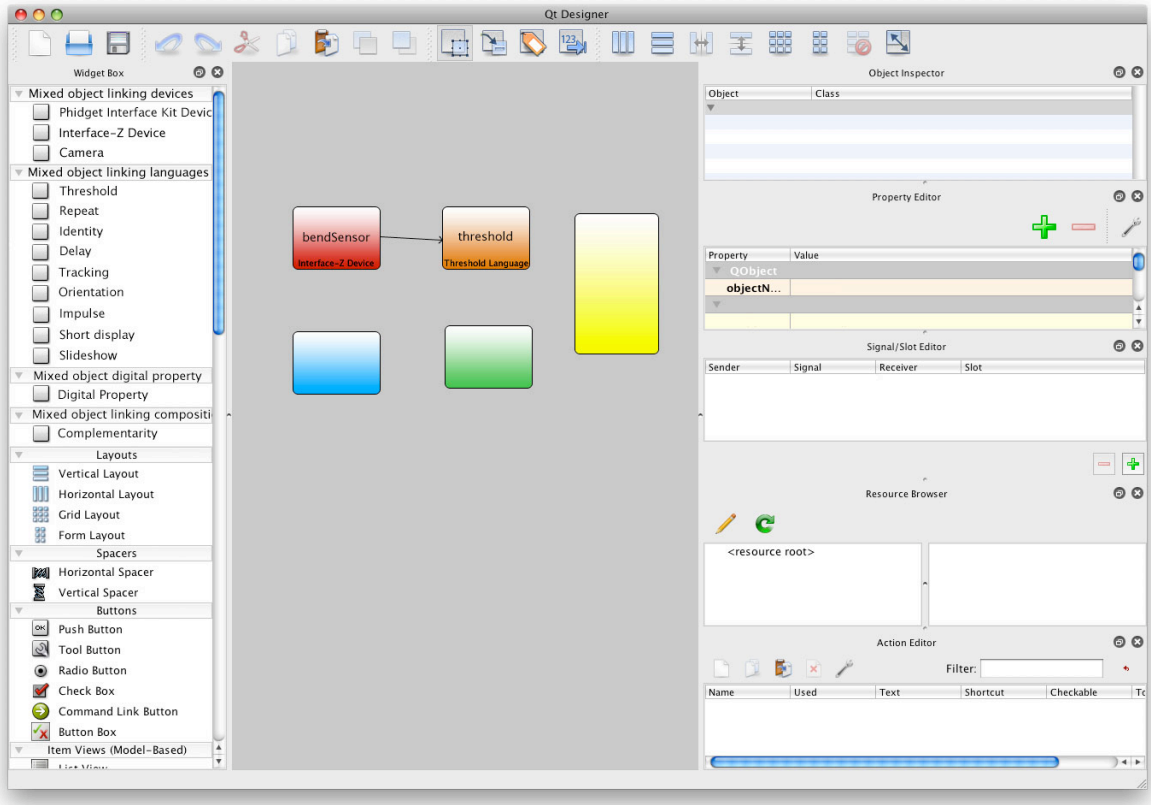


Figure 265 : Maquette d'une interface graphique (ici l'adaptation de Qt Designer 4.5) pour le prototypage d'objets mixtes.

Nous envisageons également de mettre en évidence explicitement les caractéristiques du modèle dans l'outil de prototypage des objets mixtes, afin que la transition entre activité conceptuelle et pratique soit plus fluide. Pour cela, nous pourrions par exemple expliciter les caractéristiques d'un dispositif (temporalité, nombre de dimensions, débit, résolution, etc.) dans les propriétés du composant OP. Nous pourrions également permettre à l'utilisateur de spécifier quelles caractéristiques il souhaite et lui suggérer des éléments OP qui les satisfont.

De plus, nous étudions les moyens à notre disposition pour intégrer du dynamisme dans le prototype des objets mixtes, et nous délivrer des modes édition et exécution. Pour cela, plusieurs solutions sont envisageables comme le module QtUiTools de Qt (http://doc.trolltech.com/4.5/qtuitools.html) ou QuartzComposer d'Apple par exemple.

4. Ouverture

Tout au long de ce mémoire, nous avons identifié les enjeux liés aux interfaces mixtes. Tous n'ont pas été traités dans nos travaux et pourraient constituer des pistes de recherche. Parmi ces enjeux et pour étendre nos travaux sur le long terme, nous considérons deux pistes de recherche précises : l'interaction collaborative (multi-utilisateurs) et la conception d'objets mixtes par l'utilisateur final.

4.1. Interaction collaborative

Nous avons souligné au Chapitre 2 que les interfaces mixtes sont souvent mises en avant dans la littérature pour favoriser le travail en groupe. Cet aspect des systèmes de réalité mixte n'a pas été traité dans le modèle d'interaction mixte. Pour démarrer ce travail, nous envisageons d'identifier les caractéristiques des collecticiels et de les étudier/expliciter au sein de notre modèle. Par exemple, l'observabilité publiée concerne la capacité pour un utilisateur de rendre observable des variables

d'état personnelles. Nous pourrions étudier au niveau de la modalité de liaison en sortie et/ou au niveau physique comment expliciter la caractéristique d'observabilité publiée pour un objet mixte manipulé par un utilisateur.

4.2. Conception par l'utilisateur final

Dans l'introduction du Chapitre 6, nous avons vu que les outils de prototypages sont parfois utilisés par des utilisateurs finaux. Pour permettre à l'utilisateur final de lui-même concevoir ses propres objets mixtes [Ishii et al, 2006], nous pourrions concevoir une interface mixte pour programmer des objets simples. De la même façon que les interfaces de programmation graphiques ont permis à des utilisateurs non experts de programmer, nous souhaiterions construire un atelier de construction d'objets mixtes éventuellement basé sur OP, à partir d'éléments mixtes correspondant aux concepts du modèle d'interaction mixte.

Tables des Figures

Figure 1 : Deux facettes de la conception : activités conceptuelles (idée) et pratiques (prototype) de la conception. 13
Figure 2 : Contribution conceptuelle et pratique de nos travaux. 13
Figure 3 : Structure du mémoire. 14
Figure 4: Une interface graphique, avec clavier, écran et souris. 18
Figure 5: Une interface de réalité virtuelle. Illustration extraite de http://www.limsi.fr/IMAGES2/thomas.jpg. 18
Figure 6: Un lien ou augmentation mis(e) en place grâce au poids et à une balance USB. Illustration extraite de [Carvey et al., 2006]. 19
Figure 7: Remplir le toit de tuiles avec le DigitalDesk. 20
Figure 8 : Gommer une partie des tuiles avec le DigitalDesk. 20
Figure 9 : Des utilisateurs interagissant avec le relief en agissant sur le sable. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 21
Figure 10 : Un utilisateur change de simulation en déplaçant un marqueur tangible d'un espace à un autre. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 21
Figure 11: Une visualisation de l'ensoleillement avec SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 21
Figure 12: Une visualisation de l'altitude avec SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 21
Figure 13 : L'installation matérielle du système SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 22
Figure 14 : Le système PICO. Illustration extraite de http://www.jamespatten.com/pico/picovid.html. 22
Figure 15 : PICO utilisé avec une contrainte physique. Illustration extraite de [Patten, Ishii, 2007]. 23
Figure 16 : Le système CASPER, première version (d'après [Dubois, 2001]). 24
Figure 17 : Le système CASPER, première version, et la visualisation des trajectoires de l'aiguille de ponction sur l'écran (d'après [Dubois, 2001]). 24
Figure 18: Carcade : vue de l'écran de l'ordinateur posé sur les genoux de l'utilisateur. Une caméra capture le paysage qui défile par la fenêtre et que l'utilisateur voit sur l'écran. Sur cette capture d'écran, le joueur doit passer par la fenêtre indiquée s'il ne veut pas s'écraser sur l'immeuble. 25
Figure 19 : Une vue dans le casque semi-transparent d'un joueur de l'ARQuake : on y voit à la fois le monde physique et les éléments ajoutés par l'ordinateur. 25
Figure 20: The Eye of Judgment pour PS3 (d'après http://www.eyeofjudgment.com/). 26
Figure 21: Le jeu Tuttuki Bako (d'après http://www.asovision.com/tuttuki/). 26
Figure 22: *The Golden Calf*, Jeffrey Shaw, 1994 (d'après [Shaw, 1994]). 27
Figure 23: *LiveWire*, Natalie Jeremijenko, 1995 (d'après [Jeremijenko, 1995]). 27
Figure 24: *Voice Boxes*, Natalie Jaremijenko, 1995 (d'après [Jeremijenko, 1995]). 28
Figure 25: *Alexitimia*, Paula Gaetano, 2006 (d'après [Gaetano Adi, 2006]). 28
Figure 26 : Théorie de l'action [Norman, 1986]. 29
Figure 27 : BUGbase et BUGCamera. Illustration extraite de http://buglabs.net/products. 33
Figure 28 : *Lies, all lies*, Hannes Nehls, 2003. Illustration extraite de[Nehls]. 37
Figure 29: Démarche scientifique à la fois conceptuelle et expérimentale. 39
Figure 30 : Système CASPER, première version (d'après [Dubois, 2001]). 44
Figure 31 : Visualisation des trajectoires de l'aiguille de ponction sur l'écran du système CASPER, première version (d'après [Dubois, 2001]). 44
Figure 32 : continuum de Milgram. 46
Figure 33: Exemple de configurations spatiales avec les focus de deux entités A et B (d'après [Benford, Fahlen, 1993]). A et B sont des entités et les quadrilatères de mêmes couleurs représentent l'espace ils portent leur attention (focus). 49
Figure 34: Classes de mouvements. Illustration extraite de [Benford et al., 2005]. 50

Figure 35 : Une modélisation de CASPER avec ASUR. 51
Figure 36 : Le système CASPER et la visualisation des trajectoires de l'aiguille de ponction sur l'écran (d'après [Dubois, 2001]). 53
Figure 37 : L'installation matérielle du système SandScape. Illustration extraite de http://tmg-video.media.mit.edu/sandscape/sandscape_352x240.mpg. 54
Figure 38: Modélisation de CASPER avec IRVO. 55
Figure 39: Modèle d'architecture logicielle MVC (a) et le modèle d'interaction MCRit (b). Illustration extraite de [Ullmer et al., 2005]. 56
Figure 40: Taxonomie des interfaces tangibles. Illustration extraite de [Fitzmaurice et al., 1995]. 59
Figure 41: Différents niveaux de retour d'information. Illustration extraite de [Dix et al., 2007]. 62
Figure 42: Interrupteur pour allumer la lumière. Illustration extraite de [Dix et al., 2007]. 63
Figure 43 : Bouton élastique ou à rebond contrôlant l'alimentation d'un ordinateur. Illustration extraite de [Dix et al., 2007]. 64
Figure 44 : Interrupteur pour lequel l'utilisateur doit presser suffisamment fort pour qu'il basculent dans un autre état physique. Illustration extraite de [Dix et al., 2007]. 64
Figure 45: NavRNA (a), le Phicon du grand dôme du MIT dans le *Tangible Geospace* (b), et la reacTable (c). 67
Figure 46: *Music bottles* (a), l'*Actuated workbench* utilisé avec une souris (b) ou augmenté grâce à la vision par ordinateur (c). 68
Figure 47: Un outil de la reacTable dont une propriété est contrôlée par un autre outil : Le premier est un filtre sonore passe-bas (objet bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus). Le second est un oscillateur à basse fréquence (objet circulaire et rouge avec une sinusoïde représentée dessus). 69
Figure 48 : Description générique des objets physiques (a) et numériques (b). 74
Figure 49 : Description générique d'un objet mixte. 74
Figure 50 : Description générique d'un objet mixte simple avec une modalité de liaison en entrée et une modalité de liaison en sortie, chacune étant composée d'un dispositif et d'un langage. 75
Figure 51 : L'écran modélisé comme un dispositif d'objet mixte (en haut) ou comme objet mixte (en bas). En haut, CASPER est un système de réalité augmentée pour la chirurgie. En bas, *White Snow* est une installation artistique de Jim Campbell [Campbell, 2001]. 76
Figure 52 : Exemple d'un objet mixte avec une liaison multimodale en entrée : l'aiguille de ponction du chirurgien dans CASPER [Dubois, 2001]. 77
Figure 53 : Exemple d'un objet mixte avec une liaison multimodale en sortie : une *music bottle* [Ishii et al., 2001]. 77
Figure 54 : Le modèle d'interaction instrumental appliqué à une interaction graphique : l'utilisateur navigue dans le document via la souris et une barre de défilement. 78
Figure 55 : Description générique d'une interaction simple à un haut niveau d'abstraction grâce au modèle d'interaction mixte. 79
Figure 56 : Description générique d'une interaction simple à un bas niveau d'abstraction grâce au modèle d'interaction mixte, avec le détail des objets mis en jeu dans l'interaction. 80
Figure 57 : Exemple du DigitalDesk décrit grâce au modèle d'interaction mixte : l'utilisateur gomme le dessin grâce à la gomme. 81
Figure 58 : Interaction avec le panda dans *Tuttiki Bako* (http://www.asovision.com/tuttuki/) : l'outil ne prend pas la forme d'un d'artefact, mais d'une modalité d'interaction directe. (Nous n'avons pas trouvé d'information sur la modalité en entrée pour le dispositif d'interaction.) 82
Figure 59 : Exemple d'interaction multimodale avec NavRNA [Bailly et al., 2006] : l'utilisateur manipule deux jetons pour redimensionner la molécule d'ARN projetée sur la table. 83
Figure 60 : Projection arbitraire à basse résolution pour les Music Bottles [Ishii et al., 2001]. 85
Figure 61 : Projection d'un point vert à haute résolution pour l'Actuated Workbench. 85
Figure 62 : ReacTable [Jordà et al., 2007]. 86
Figure 63 : Tangible geospace [Ullmer, Ishii, 1997], avec deux objets mixtes : le Phicon et la carte. 86
Figure 64 : Du retard dans la liaison de l'objet manipulé avec l'*actuated workbench*. 89
Figure 65 : Caractérisation intrinsèque des propriétés physiques d'un objet mixte selon leur prise en compte par une modalité de liaison en entrée ou en sortie. 90
Figure 66 : NavRNA [Bailly et al., 2006]. 90

Figure 67 : *Actuated workbench* utilisé avec une souris : la position du palet n'est pas capturée par une caméra. 91
Figure 68: Les objets de PICO ont une position physique d'équilibre. Illustration extraite de http://www.jamespatten.com/pico/picovid.html 92
Figure 69 : Caractérisation intrinsèque des propriétés numériques d'un objet mixte selon leur prise en compte par une modalité de liaison en entrée ou en sortie. 93
Figure 70 : L'*actuated workbench* utilisé avec la trackball. 93
Figure 71 : Description de l'interaction avec l'*actuated workbench*, à l'aide d'une trackball. 94
Figure 72 : Remplir le toit de tuiles avec le DigitalDesk. 95
Figure 73 : Gommer une partie des tuiles avec le DigitalDesk. 95
Figure 74 : « Mets ça là ». Illustration extraite de [Bolt, 1980]. 96
Figure 75 : Exemple de caractérisation de la métaphore de nom avec le modèle d'interaction mixte : le Tangible Geospace et sa métaphore se rapportant au paramètre de la tâche. 98
Figure 76 : Exemple de caractérisation de la métaphore de verbe avec le modèle d'interaction mixte: une *music bottle*. 99
Figure 77 : Typologie des ports d'un objet mixte 100
Figure 78 : ReacTable : Un oscillateur à basse fréquence (objet 2, circulaire et rouge avec une sinusoïde représentée dessus) qui contrôle un filtre sonore passe-bas (objet 1, bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus). 101
Figure 79 : Classification des exemples : Chaque système appartient à une unique catégorie. Dans ce diagramme, par souci de clarté, nous ne montrons qu'une différence basée sur une caractéristique entre chaque système, bien que plusieurs caractéristiques nous permettent de les distinguer. 103
Figure 80 : Espace d'évaluation des caractéristiques du modèle. 104
Figure 81 : RAZZLE, version existante. Un enfant joue à la fête de la science et ramasse une pièce avec la modalité gestuelle. 115
Figure 82 : Vue de ce que l'utilisateur voie dans les lunettes semi-tansparentes de RAZZLE, première version (les pixels noirs sont transparents). 115
Figure 83 : RAZZLE, première version. L'utilisateur porte le matériel nécessaire, et un compère joue le rôle du Magicien d'Oz (au premier plan, à gauche). 116
Figure 84 : Description d'une pièce de puzzle dans RAZZLE selon le modèle d'interaction mixte. 117
Figure 85 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE, première version : Ramasser une pièce en s'approchant à moins d'un mètre pendant quelques secondes (M1). 118
Figure 86 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE : Ramasser une pièce en s'approchant et l'attrapant d'un geste de la main (M2). 119
Figure 87 : Modalité proposée pour ramasser une pièce de puzzle dans la première version de RAZZLE : Ramasser une pièce en positionnant le curseur de la souris dessus et en cliquant (M3). 120
Figure 88 : Interface du magicien d'oz. 121
Figure 89 : Cadre pour la conception de RAZZLE. 123
Figure 90 : Modalité tactile. 124
Figure 91 : Un joueur de RAZZLE porte l'équipement nécessaire (des lunettes semi-transparentes avec un capteur d'orientation monté dessus, un touchpad, ici dans sa main, et un ordinateur portable dans le sac à dos). 125
Figure 92 : Modalité vocale. 126
Figure 93 : Modalité gestuelle. 127
Figure 94 : Un utilisateur ramasse une pièce trouvée grâce à la modalité gestuelle. 127
Figure 95 : Une joueuse est à la recherche des pièces à trouver pendant l'évaluation ergonomique de RAZZLE. 128
Figure 96 : Taux d'utilisation des modalités pour toutes sessions et sujets confondus. 128
Figure 97 : Taux d'utilisation des modalités par sujet (toutes commandes et sessions confondues). 129
Figure 98 : Comparaison de l'usage des modalités par sujet entre la première et la troisième session (toutes actions confondues). 129

Figure 99 : Taux d'utilisation des modalités pour une action donnée (toutes sessions et sujets confondus). 130
Figure 100 : Taux d'utilisation des modalités pour chaque action et chaque session. 130
Figure 101 : Taux d'utilisation des modalités pour chaque action et session par le sujet 4. 131
Figure 102 : Taux d'utilisation des modalités pour chaque action et session par le sujet 9. 131
Figure 103 : Vue de l'installation interactive *Playground* à la Gallerie DuBellay, Mont Saint-Aignan, Seine-Maritime, exposée en mars 2007 lors de l'exposition intitulée *Groupe*. 132
Figure 104 : Schéma décrivant le parc augmenté selon le modèle d'interaction mixte. 133
Figure 105: Un exemple de parcours du jeu Snap2Play sur le campus de la National University Singapore. Les points bleus (D) sur le parcours indique une carte numérique, et les points (P) indique une carte physique. 134
Figure 106 : Captures d'écrans du téléphone portable dans les différentes phases du jeu Snap2Play. 134
Figure 107 : Point de départ pour concevoir des cartes de jeu de Memory similaires dans Snap2Play. 135
Figure 108 : Copier les propriétés physiques et numériques pour concevoir des cartes de jeu de Memory similaires dans Snap2Play. 136
Figure 109 : Copier les propriétés physiques et numériques nécessaires pour concevoir des cartes de jeu de Memory similaires dans Snap2Play. 137
Figure 110 : Concevoir une modalité de liaison en entrée pour les cartes de jeu de Memory dans Snap2Play. 138
Figure 111 : Concevoir une modalité de liaison en sortie pour les cartes de jeu de Memory dans Snap2Play : rendre observable l'état de la carte. 139
Figure 112 : Concevoir une modalité de liaison en sortie pour les cartes de jeu de Memory dans Snap2Play : rendre observable l'image de référence de la carte. 140
Figure 113 : Conception finale retenue pour les cartes de jeu de Memory dans Snap2Play. 140
Figure 114 : Ramasser une carte de Memory par la parole. 141
Figure 115 : Ramasser une carte de Memory en « prenant une photo » : il s'agit de viser et appuyer sur un bouton. 141
Figure 116 : Dispositifs d'interaction avec les cartes de Memory dans Snap2Play. 142
Figure 117 : Résultats de l'enquête qui a suivi les expériences de jeu avec Snap2Play. 143
Figure 118 : Prototypes de haute fidélité en silicone (non interactif) du système ORBIS. 143
Figure 119 : Quelle que soit l'orientation de l'objet dans le plan vertical, la photo affichée est toujours dans le bon sens. 144
Figure 120 : Objet de la tâche d'ORBIS : première étape de conception. 144
Figure 121 : Objet de la tâche d'ORBIS : les propriétés numériques identifiées dans la première étape de conception ne sont pas acquises, mais sont matérialisées. 144
Figure 122 : Exploration des possibilités de conception pour la modalité de liaison en sortie. 145
Figure 123 : Conception des propriétés physiques de l'objet de la tâche dans ORBIS. 146
Figure 124 : Concevoir une modalité de liaison en entrée pour lier l'orientation de l'objet avec celle des photos. 146
Figure 125 : Conception finale de l'objet de la tâche d'ORBIS : la liste de photos mixte. 146
Figure 126 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « démarrer/arrêter la présentation ». 148
Figure 127 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « mélanger les photos ». 150
Figure 128 : Schéma de l'interaction entre l'utilisateur et la liste de photos augmentée pour la tâche « naviguer dans les photos ». 151
Figure 129 : Deux contextes différents pour l'utilisation d'ORBIS : tenu dans la main (gauche) ou posé sur une table (droite). 152
Figure 130 : Le prototype final de l'outil mixte dans Roam (Figure 131) dans la main de l'utilisateur. 153
Figure 131 : Le prototype final de l'outil mixte dans Roam, avec capteur de flexion et diodes électroluminescentes vertes et jaunes. 154

Figure 132: Alternatives de conception pour les ports de sorties physiques : le son avec l'exemple du haut-parleur (à gauche), la lumière (ici avec l'exemple des LEDs (au centre et à droite). En considérant les ports de sortie, physiques, nous constatons que selon la façon dont l'outil est pris en main par l'utilisateur, la lumière peut être cachée par sa main (au centre) ou non (à droite).....156
Figure 133 : Point de départ de conception intrinsèque de l'outil d'enregistrement de Roam. 156
Figure 134 : Autre prototype de l'outil pour Roam, avec des propriétés physiques spatialement séparées.....157
Figure 135 : Marqueurs RFID fournis du système Yubi posés par un utilisateur sur un objet personnel, souvenir de vacances. 158
Figure 136 : Interface graphique pour accéder aux données associées à un objet : nous observons que 17 éléments numériques sont associés à cet objet mixte. L'accès à liste de ces éléments se fait en cliquant sur la flèche (en bas à droite). Pour ajouter des éléments, il suffit de les glisser-déposer dans la zone ronde centrale. 158
Figure 137: Description d'un objet augmenté avec Yubi selon le modèle d'interaction mixte.....158
Figure 138 : Document produit par le designer du système Yubi : Description d'un objet mixte et entouré d'annotations.....159
Figure 139 : Arbre de classification des véhicules, remplaçant les espèces pour l'expérience de conception en groupe focalisé.....161
Figure 140 : La proposition initiale à partir de laquelle les participants au groupe focalisé ont conçu de nouvelles techniques d'interaction. Ce prototype propose deux points de vue de l'arbre : une vue égocentrique sur l'écran et une vue exocentrique sous forme papier. 162
Figure 141 : Exactitude de la réponse pour le modèle d'interaction mixte : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés. 166
Figure 142 : Exactitude de la réponse pour le modèle d'interaction mixte selon le profil des sujets : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés. 166
Figure 143 : Exactitude de la réponse pour ASUR selon le profil des sujets : Proportion des concepts attendus bien identifiés, non précis, mal identifiés ou non identifiés. 167
Figure 144 : Facilité moyenne (subjective) selon les profils (1 signifie très difficile, 4 signifie très facilement). 167
Figure 145 : Temps moyen (subjectif) selon le profil et le modèle (1 signifie très lentement, 4 signifie très rapidement).....168
Figure 146 : Confiance moyenne accordée à la réponse (1 signifie pas confiant du tout, 10 signifie tout à fait confiant).....168
Figure 147 : Moyenne des notes de compréhensibilité des schémas par tous les sujets pour le modèle d'interaction mixte et pour ASUR.....169
Figure 148 : Effectif des participants en fonction des notes de confiance attribuées (10 signifie tout à fait confiant et 1 pas du tout confiant). 169
Figure 149 : Effectif des participants en fonction des notes de compréhensibilité attribuées (10 signifie tout à fait compréhensible et 1 pas du tout compréhensible). 170
Figure 150 : Moyenne par profil de l'aide apportée par les choix graphiques des deux notations : 1 signifie « n'aide pas du tout », 4 signifie « aide tout à fait ».....170
Figure 151 : Proportion de bonnes réponses aux sept questions de compréhension des éléments du modèle d'interaction mixte.....171
Figure 152 : La Partie I du mémoire présente notre contribution concernant l'exploration conceptuelle de l'espace des possibilités (à gauche, le modèle d'interaction mixte) et la Partie II présente notre contribution concernant l'exploration pratique de l'espace des possibilités (à droite, l'outil logiciel OP). 174
Figure 153 : Prototypage simultané mais séparé de l'apparence et de l'interaction. Illustration extraite de [Hudson, Mankoff, 2006].....179
Figure 154 : Interfaces matérielles Arduino : (a) l'interface USB de base (Duemilanove), (b) Lily Pad cousue dans du tissu, et (c) l'Arduino mini. Illustrations extraites de http://www.arduino.cc/..181
Figure 155 : Exemple de kit de Phidgets pour débuter : (de haut en bas et de gauche à droite) un capteur de toucher, une glissière, un capteur de lumière, un potentiomètre, un capteur de force,

l'interface USB sur laquelle brancher les capteurs et effecteurs, 8 interrupteurs, et des LEDs. Illustration extraite de http://www.phidgets.com. 182
Figure 156 : Bouton de Switcharoo à épingler sur un prototype grâce à ses trois épingles. Illustration extraite de [Avrahami, Hudson, 2002]. 183
Figure 157 : Prototype fait de mousse pour explorer l'apparence et équipé de dispositifs de Calder pour explorer l'interaction. Illustration extraite de [Lee et al., 2004]. 183
Figure 158: Dispositifs matériels filaires de Calder : (A) carte d'entrée/sortie, (B) ensemble de 4 boutons, (C) manette analogique, (D) simple bouton, (E) composant d'entrées numériques/analogiques, (F) potentiomètre rotatif. Illustration adaptée de [Lee et al., 2004]... 184
Figure 159: Dispositifs matériels sans fils de Calder : (A) potentiomètre rotatif, (B) ensemble de 4 boutons, (C) transmetteur sans fil, (D) capteur d'inclinaison à deux axes, (E) ensemble de trois LEDs. Illustration adaptée de [Lee et al., 2004]. 184
Figure 160 : BOXES, un outil pour combiner le prototypage à basse résolution de l'apparence et d'interaction. Illustration extraite de la vidéo de [Hudson, Mankoff, 2006]. 185
Figure 161 : Exemple d'application construite avec l'ARTolKit. 186
Figure 162 : AiRtoolkit : Extension de l'ARToolKit pour reconnaître de nouveaux types de marqueurs (à droite). L'utilisateur peut dessiner ses propres marqueurs sur un papier pré-quadrillé (en haut à gauche) et les prendre en photo (en haut au centre) pour les lier à un élément numérique, comme une étoile en 3D, ou à une fonction, comme allumer la lumière (en bas au centre). De nombreuses formes sont possible (en bas à gauche). Illustrations extraites de http://sketchblog.ecal.ch/variable_environment/. 187
Figure 163 : Ligne de code écrite avec Papier-Mâché pour créé un VisionPhob. 188
Figure 164 : Interface pour le développeur travaillant avec Papier-Mâché (partie de gauche). Illustration extraite de [Klemmer et al., 2004]. 188
Figure 165 : Interface pour le développeur travaillant avec Papier-Mâché (partie de droite). Illustration extraite de [Klemmer et al., 2004]. 189
Figure 166 : Ligne de code écrite avec Papier-Mâché pour associer les Phobs avec des éléments numériques. 189
Figure 167 : iRoom de Stanford University. 190
Figure 168 : Dispositifs iStuff. Illustration extraite de [Ballagas et al., 2003]. 191
Figure 169 : Architecture d'iStuff. Illustration extraite de [Ballagas et al., 2003]. 191
Figure 170 : Architecture d'iStuff mobile. Illustration extraite de [Ballagas et al., 2007]. 192
Figure 171 : d.tools pendant son utilisation. L'éditeur graphique est affiché sur l'écran de l'ordinateur, et le prototype sur la table est branché à l'ordinateur. 194
Figure 172 : Dispositifs matériels intégrés dans d.tools. Illustration extraite de http://hci.stanford.edu/dtools/. 195
Figure 173 : Interface graphique de d.tools pour la conception. Illustration extraite de [Hartmann et al., 2006]. 195
Figure 174 : Évaluation d'un prototype avec d.tools. Illustration extraite de [Hartmann et al., 2006]. 196
Figure 175 : Interface graphique de d.tools pour comparer les données expérimentales de quatre évaluations. Illustration extraite de [Hartmann et al., 2006]. 196
Figure 176 : Exemplar : Démonstration de l'interaction pour avoir des données exemples, éditer pour corriger ces données exemples, puis vérifier que ces corrections sont adéquates. Illustration extraite de [Hartmann et al., 2007]. 197
Figure 177 : Editeur de code de Juxtapose : l'exemple de deux alternatives pour la visualisation d'une carte. Une ligne de code est surlignée en bleu, signifiant que cette ligne diffère selon les prototypes. La solution alternative décrite consiste ici à fixer la luminosité à 100. Illustration extraite de [Hartmann et al., 2008b]. 198
Figure 178 : Environnement d'exécution des alternatives dont une partie du code est présenté Figure 177. Par exemple, parmi les différences entre l'espace de travail de gauche et celui du milieu, nous trouvons la luminosité de la carte (Figure 177). À ce paramètre s'ajoutent d'autres paramètres, présentés et contrôlables dans l'espace de travail de droite de l'interface. Illustration extraite de [Hartmann et al., 2008b]. 198

Figure 179 : Fragment de code montrant comment la variable « brightness » pourra être ajustée pendant l'exécution, et comment la variable « counter » ne le sera pas. 199
Figure 180 : Interface d'ICON (a) pour une interaction bi-manuelle (b). 200
Figure 181 : Exemple d'interaction prototypée avec la plateforme OpenInterface. Illustration extraite de [Serrano et al., 2008]. 201
Figure 182 : Modèle ASUR d'un système de réalité mixte pour la construction d'un arbre de classification des espèces. Illustration extraite de [Gauffre et al., 2007]. 202
Figure 183 : Le modèle ASUR-IL correspondant au modèle ASUR de la Figure 182. Illustration extraite de [Gauffre et al., 2007]. 202
Figure 184 : Mécanisme des signaux et des slots de Qt. Illustration extraite de http://doc.trolltech.com/4.4/signalsandslots.html. 211
Figure 185 : Structure de la boîte à outils OP : diagramme d'héritage des composants OP. 214
Figure 186 : Modélisation d'un objet mixte pliable. 216
Figure 187 : Construction de l'interface graphique pour la mise au point de l'objet pliable modélisé à la Figure 186. 216
Figure 188 : Interface graphique de mise au point d'un objet : exemple d'un objet appelé *Object* qui capte la flexion parmi ses propriétés physiques, et met sa propriété numérique *MyProperty* à vrai ou faux selon que le capteur de flexion envoie une valeur respectivement au dessous ou au dessus de 50. Comme retour d'information, cet objet fournit un message affichant la valeur de *MyProperty* (0 ou 1) sur un écran. 217
Figure 189 : Carte matérielle d'Interface-Z pour les capteurs. Illustration adaptée de http://www.interface-z.com/. 219
Figure 190 : Carte matérielle d'Interface-Z pour les effecteurs : 2 servomoteurs et 4 autres actionneurs, comme des lampes par exemple. Illustration adaptée de http://www.interface-z.com/. 219
Figure 191 : Pour utiliser un dispositif branché à la carte matérielle de capteurs (Figure 189) ou d'effecteurs (Figure 190). 220
Figure 192 : Exemple de prototypage avec un capteur de flexion interface-Z : configuration matérielle pouvant correspondre au code *MIDIDevice BendSensor("BendSensor", LinkingComponent::IN, Index, isAnalogue, Resolution);* avec Index = 0 (place du branchement, entouré en rouge), isAnalogue = true et Resolution = 7. 221
Figure 193 : Phidget Interface Kit 8/8/8. Illustration adaptée de http://www.phidgets.com/. 221
Figure 194 : Exemples de capteurs analogiques des Phidgets : (de haut en bas et de gauche à droite) un capteur de toucher, une glissière, un capteur de luminosité, un capteur de force. Illustration extraite de http://www.phidgets.com/. 222
Figure 195 : Exemples de capteurs numériques des Phidgets : des interrupteurs. Illustration extraite de http://www.phidgets.com/. 222
Figure 196 : Exemples d'effecteurs numériques : des diodes électroluminescentes. Illustration extraite de http://www.phidgets.com/. 222
Figure 197 : Pour utiliser un dispositif branché à l'interface Kit 8/8/8 des Phidgets. 223
Figure 198 : Exemple de prototypage avec une glissière physique des Phidgets et une diode électroluminescente branchée à une sortie numérique de la même carte : configuration matérielle pouvant correspondre au code *PhidgetInterfaceKitDevice Slider("Slider", LinkingComponent::IN, SliderIndex, isSliderAnalogue); PhidgetInterfaceKitDevice LED("LED", LinkingComponent::IN, LEDIndex, isLEDAnalogue);* avec SliderIndex = 1 (entouré en rouge), LEDIndex = 0 (entouré en bleu), isSliderAnalogue = true, isLEDAnalogue = false. 223
Figure 199: Pour utiliser la reconnaissance des marqueurs de l'ARToolKit. 224
Figure 200 : Constructeur pour créer un *OrientationInputLanguage*. 225
Figure 201 : Transformation effectuée par le composant *IdentityInputLanguage*. 225
Figure 202 : Constructeur pour créer un *IdentityInputLanguage*. 226
Figure 203 : Transformation effectuée par le composant *RampInputLanguage*. Si *LowerThreshold* vaut *Min* et *UpperThreshold* vaut *Max*, alors nous retrouvons la même transformation que l'identité (Figure 201). 227
Figure 204 : Constructeur pour créer un RampInputLanguage. 227
Figure 205 : Transformation effectuée par le composant ThresholdInputLanguage. 227

Figure 206 : Pour créer un langage de liaison « seuil ». 227
Figure 207 : Pour créer un langage de liaison « délai ». 227
Figure 208 : Pour créer un langage de liaison « répétition ». 228
Figure 209 : Exemple d'affichage d'une image par le composant *ShortDisplayOutputLanguage* dans le cas de l'outil de lecture pour ORBIS. (Chapitre 5, section 3.1.2.1.2.1). 228
Figure 210 : Pour créer une composition entre deux modalités de liaison. 229
Figure 211 : Pour créer un composant DigitalProperty, en spécifiant le type de la propriété avec une chaîne de caractères. 230
Figure 212 : Pour créer un composant DigitalProperty, en spécifiant le type de la propriété avec un type QVariant. 230
Figure 213 : Exemple d'un prototype d'objet sensible à la lumière. 231
Figure 214 : Description de l'objet sensible à la lumière de la Figure 213, selon le modèle d'interaction mixte. 231
Figure 215: Code utilisant la boîte à outils OP pour prototyper l'objet de la Figure 213. 232
Figure 216 : Code utilisant l'outil de construction d'interface graphique associé à l'objet codé à la Figure 215. L'interface graphique générée est présentée Figure 217. 232
Figure 217 : Interface graphique pour la mise au point de l'objet présenté à la Figure 213 et prototypé à la Figure 215. Le code générant cette interface est présenté à la Figure 216. 233
Figure 218 : Faire clignoter la diode électroluminescente quatre fois, à partir du code de la Figure 215. 233
Figure 219 : Utiliser un capteur de luminosité d'Interface-Z, à la place du capteur des Phidgets de la Figure 215. 233
Figure 220: Slot écrit dans un *QObject* simple servant d'interface entre l'application Google Earth (http://earth.google.com/) et l'objet sensible à la lumière de la Figure 213. L'objet peut ainsi servir à tourner la terre dans Google Earth. CGPostKeyboardEvent est une fonction du ApplicationServices framework d'Apple qui permet de simuler des évènements clavier. 235
Figure 221 : Démarche de conception de l'outil OP. 253
Figure 222: Prototype matériel de l'objet de la tâche, développé avec OP. 254
Figure 223: Code OP correspondant au prototype de la Figure 222. 255
Figure 224: Prototype physique final de l'outil pour naviguer dans la liste de photos. 255
Figure 225: Code OP l'outil pour naviguer dans la liste de photos. 256
Figure 226: Prototypage de l'outil avec une modalité de liaison captant la rotation. 256
Figure 227: Prototypage de l'outil avec une modalité de liaison captant la pression. 256
Figure 228: Prototypage de l'outil avec un langage identité pour la liaison en entrée. 257
Figure 229: Prototypage de l'outil avec un langage rampe pour la liaison en entrée. 257
Figure 230 : Prototypage de la modalité de liaison en sortie avec un message qui s'affiche à l'écran : code OP (haut) et fenêtres affichées pendant 1 seconde par dessus l'objet de la tâche (bas, de gauche à droite) lorsqu'on affiche les photos du prototype de la première (à gauche) à la quatrième et dernière (à droite). 257
Figure 231: Prototypage de la modalité de liaison en sortie avec un haut-parleur : code OP (gauche) et haut parleur utilisé inséré dans le prototype de l'outil (droite). 258
Figure 232 : Prototype physique de l'outil de démarrage/arrêt de la présentation avec une modalité de liaison captant la flexion. 258
Figure 233 : Prototype physique de l'outil de démarrage/arrêt de la présentation avec une modalité de liaison captant la pression. 258
Figure 234 : Code OP de l'outil pour démarrer/arrêter la présentation des photos. 259
Figure 235 : Images affichées lorsque la présentation des photos est démarrée (gauche) ou arrêtée (droite). 259
Figure 236 : Code OP de l'outil pour mélanger les photos. 260
Figure 237 : Image affichée lorsque l'outil dédié au mélange des photos est secoué (« RDM » signifie *Random*). 260
Figure 238 : Code OP complet de l'outil d'enregistrement pour Roam. 262
Figure 239 : Alternatives physiques et logicielles pour les dispositifs de liaison en entrée pour l'outil d'enregistrement de Roam : capteur d'étirement d'Interface-Z (haut), potentiomètre des Phidgets (centre), et capteur de luminosité des Phidgets (bas). 263

Figure 240: Alternatives de conception : sortie sonore avec un haut-parleur (à gauche), lumineuse avec une LED (à droite). 263
Figure 241 : Alternatives utilisant le composant *BeepOutputLanguage* avec le haut-parleur pour l'outil d'enregistrement de Roam. 263
Figure 242 : Alternative utilisant le composant *BeepOutputLanguage* avec la diode pour l'outil d'enregistrement de Roam, afin de matérialiser les propriétés *isOn* et *isOk*. 264
Figure 243 : Alternatives utilisant le composant *BeepOutputLanguage* avec une répétition (composant *RepeatLanguage*) avec la diode pour la propriété *isOn* de l'outil d'enregistrement de Roam. 264
Figure 244: Prototype physique complet de l'outil dans Roam (correspondant au code OP de la Figure 238) : dans la main de l'utilisateur (gauche) et en gros plan (droite), avec capteur de flexion et diodes électroluminescentes, une verte et une jaune. 264
Figure 245 : Capture d'écran de l'enquête auprès des personnels du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé. 268
Figure 246 : Représentation graphique de l'expertise en C++ des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 8 n'ayant aucune connaissance, 5 novices, 20 peu expérimenté(e)s, 29 plutôt expérimenté(e)s, 7 experts. 269
Figure 247 : Représentation graphique de l'expertise en Qt des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 50 n'ayant aucune connaissance, 9 novices, 6 peu expérimenté(e)s, 2 plutôt expérimenté(e)s, 2 experts. 269
Figure 248 : Représentation graphique de l'expertise en Phidgets des participants à l'enquête auprès du personnel du Laboratoire d'Informatique de Grenoble pour recruter des participants au groupe focalisé : 65 n'ayant aucune connaissance, 1 novice, 1 peu expérimenté(e), aucun plutôt expérimenté(e), aucun expert. 270
Figure 249: Expérience pilote avec des novices en C++, Qt et Phidgets (peu ou aucune connaissance de ces outils). Au premier plan, deux participants et à l'arrière-plan, une expérimentatrice. 270
Figure 250 : Descriptions d'un même objet mixte fournies pour le premier exercice aux participants de la première expérience. 271
Figure 251 : Code OP fourni aux participants du groupe focalisé pour le 2ème exercice. 271
Figure 252 : Code OP à modifier. 272
Figure 253 : Première modification à effectuer (Chaque modification étant tirée au sort par les participants). 272
Figure 254 : Deuxième modification à effectuer (Chaque modification étant tirée au sort par les participants). 273
Figure 255 : Schéma réalisé par un participant de la première expérience pour expliquer le code de la Figure 251. 273
Figure 256 : Expérience avec des informaticiens connaissant Qt. À gauche, manipulation du prototype en bleu. À droite, recherche d'information dans la documentation. 278
Figure 257 : Changements effectués dans la description textuelle de l'objet mixte pour la première expérience. 278
Figure 258 : Code OP à modifier lors de la seconde expérience. 279
Figure 259 : Schémas réalisés par un participant de la deuxième expérience pour expliquer le code de la Figure 251. 280
Figure 260 : Schéma réalisé par un participant de la deuxième expérience pour expliquer le code de la Figure 251. 281
Figure 261 : Schéma réalisé par un participant de la deuxième expérience pour expliquer le code de la Figure 251. 281
Figure 262 : Schéma réalisé par un participant de la deuxième expérience avant d'écrire le pseudo-code à partir du texte de la Figure 257. 282
Figure 263 : Évaluations de l'outil OP. 285
Figure 264 : Jeux de cartes à utiliser pour la conception (d'après Alexandre Elmir, http://www.davectors.com/) 291
Figure 265 : Maquette d'une interface graphique (ici l'adaptation de Qt Designer 4.5) pour le prototypage d'objets mixtes. 292

Figure 265: Exemple d'interaction avec le DigitalDesk : l'utilisateur place (a) et appuie sur (b) le bouton FILL, pour remplir le dessin du toit de la maison par des tuiles (c).....318
Figure 266: Description d'une interaction avec le DigitalDesk : l'utilisateur place et appuie sur le bouton FILL, pour remplir le dessin du toit de la maison par des tuiles. 318
Figure 267: Exemple d'interaction avec Sandscape : l'utilisateur place le marqueur sur un emplacement prévu, pour choisir une simulation. 319
Figure 268: Description de l'interaction avec Sandscape. 319
Figure 269: Interaction avec le palet de l'*actuated workbench*, à l'aide d'une trackball. 320
Figure 270: Description de l'interaction avec l'*actuated workbench*, à l'aide d'une trackball. 320
Figure 271: Interaction avec le palet de l'*actuated workbench* en mode vision par ordinateur. 321
Figure 272: Description de l'interaction avec l'*actuated workbench*, en mode vision par ordinateur. Il n'y a pas de tâche, donc de notions d'outil ou objet de la tâche dans ce scénario. Cet objet mixte pourrait être l'un ou l'autre, selon l'application. 321
Figure 273 : Le système PICO (illustration issue de http://www.jamespatten.com/pico/picovid.html). 321
Figure 274 : Description de l'interaction avec deux palets de PICO. 322
Figure 275: Interaction avec CASPER (Computer ASsisted PERicardial surgery). 322
Figure 276: Description de l'interaction avec CASPER, première version. Il n'y a pas ici de langage d'interaction entre les deux objets : il s'agit d'un système de réalité augmente où seule l'évaluation est augmentée. 323
Figure 277 : Installation de Carcade dans une voiture : un camera sur la vitre capte l'image du décor qui défile (a), et l'utilisateur manipule l'avion grâce à une manette de jeu et voit l'avion et le paysage sur son écran (b). 323
Figure 278: Interaction avec Carcade : des éléments numériques, comme les étoiles bleues, sont ajoutés au décor. 324
Figure 279: Interaction avec Carcade : Un élément du décor physique (immeuble) est reconnu et barre le chemin de l'avion. Le système ajoute un passage numérique (a) (en vert). Si le joueur ne présente pas à temps son avion en face du passage, l'avion s'écrase (b). 324
Figure 280: Description de l'interaction avec Carcade : La manette n'est pas détaillée faute d'informations. 325
Figure 281: Description de l'interaction avec l'installation interactive *The Golden Calf* [Shaw, 1994]. 326
Figure 282: *Alexitimia*, Paula Gaetano, 2006 (d'après [Gaetano Adi, 2006]).....326
Figure 283 : Description de l'interaction avec Alexitimia [Gaetano Adi, 2006]. 327
Figure 284: *Voice Boxes*, Natalie Jaremijenko, 1995 (d'après [Jeremijenko, 1995]). 327
Figure 285 : Description de l'interaction pour la lecture du contenu sonore de la *voice box* [Jeremijenko, 1995]. 328
Figure 286 : Description de l'interaction pour l'enregistrement du contenu sonore de la *voice box* [Jeremijenko, 1995]. 329
Figure 287: *LiveWire*, Natalie Jeremijenko, 1995 (d'après [Jeremijenko, 1995]). 329
Figure 288 :Description de l'interaction avec le *LiveWire* [Jeremijenko, 1995]. 330
Figure 289: The Eye of Judgment pour PS3 (d'après http://www.eyeofjudgment.com/). 330
Figure 290 : Description d'une interaction avec une carte de The Eye of Judgment. Il n'y pas de langage d'interaction car il s'agit ici de réalité augmentée où seule l'évaluation de l'objet est augmentée.....331
Figure 291: Un outil de la reacTable dont une propriété est contrôlée par un autre outil : Le premier est un filtre sonore passe-bas (objet bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus). Le second est un oscillateur à basse fréquence (objet circulaire et rouge avec une sinusoïde représentée dessus). 331
Figure 292 : Description d'une interaction avec la reacTable. 332
Figure 293 : Une vue dans le casque semi-transparent d'un joueur de l'ARQuake [Thomas et al., 2000] : on y voit à la fois le monde physique et les éléments ajoutés par l'ordinateur. 332
Figure 294 : Description d'une interaction avec ARQuake [Thomas et al., 2000]. 333
Figure 295 : Description d'une solution de conception pour les bandes de papier augmentées du système pour les contrôleurs aériens [Mackay et al., 1998] : ce qui est écrit sur la bande de

papier sa position sont capturés avec une tablette graphique. L'information associée à la bande de papier est affichée sur un écran. 334
Figure 296 : Premières idées générées pour l'objet de la tâche. 336
Figure 297 : Objet de la tâche sous forme de cornes d'animal. 337
Figure 298 : Une barre d'onglets pour avancer dans l'arbre. 338
Figure 299 : Variante de la barre d'onglet pour avancer dans l'arbre : deux modalités équivalentes sont proposées en entrée, au cas où la métaphore ne serait pas comprise. 339
Figure 300 : Onglet pour reculer d'un niveau dans l'arbre. 340
Figure 301 : Illustration de la composition spatiale proposée : Dans le coin supérieur droit est affichée l'arbre (objet de la tâche, Figure 296) et le critère parent (outil pour reculer, Figure 300). Au milieu est afficher le critère courant (objet de la tâche). En bas ou a l'horizontal se trouve l'outil pour reculer (Figure 300) juxtaposé avec l'outil pour avancer (Figure 298). 341
Figure 302 : Outil onglet pour se déplacer latéralement dans l'arbre (frère de gauche/droite). 342
Figure 303 : Début de solution pour un outil pour se déplacer latéralement dans l'arbre (frère de gauche/droite).) 343
Figure 304 : Début de conception d'outil pour avancer dans un critère, reculer, et voir le frère de gauche ou droite du critère courant. 344
Figure 305 : Début de solution pour un outil pour avancer ou reculer. 345
Figure 306 : Outil pour construire l'arbre, objet de la tâche de la Figure 297. 346
Figure 307 : Transformation effectuée par le composant *RampInputLanguage*. 384
Figure 308 : Constructeur pour créer un RampInputLanguage. 384
Figure 309 : Transformation effectuée par le composant *ThresholdInputLanguage*. 385
Figure 310 : Pour créer un langage de liaison « seuil ». 385
Figure 311 : Pour créer un langage de liaison « délai ». 386
Figure 312 : Pour créer un langage de liaison « répétition ». 386
Figure 313 : Pour créer un langage de liaison en sortie *ShortDisplay*. 387
Figure 314 : Pour créer un langage *Beep*. 387

Publications

Article de journaux internationaux, avec comité de lecture

Chin, You, Coutrix, Lim, Chevallet, Nigay, Snap2Play, *A Mixed-Reality Game based on Scene Identification*, MMM'08 Special Issue of "The Visual Computer" journal

Chapitres de livre

Coutrix, Nigay, *An Integrating Framework for Mixed Systems*, The Engineering of Mixed Reality, Springer, 2009

Nigay, Coutrix, Renevier, *Systèmes interactifs mixtes : Fusion des mondes physique et numérique*, Interfaces numériques (Collection information, hypermédias et communication), Chapitre 3, Hermès Science, ISBN13 978-2-7462-1695-2, 18 pages, 2007.

Articles longs, conférences internationales avec comité de lecture

Chin, You, Coutrix, Lim, Chevallet, Nigay, *Snap2Play: A Mixed-Reality Game based on Scene Identification*, Proceedings of the 14th international multimedia modeling conference (MMM'08)

Coutrix, Nigay, *Mixed Reality: A Model of Mixed Interaction*, Proceedings of the 8th International Conference on Advanced Visual Interfaces (AVI'06)

Articles courts, conférences internationales avec comité de lecture

Coutrix, Nigay, *Balancing Physical and Digital Properties in Mixed Objects*, Proceedings of the 9th International Conference on Advanced Visual Interfaces (AVI'08)

You, Chin, Lim, Chevallet, Coutrix, Nigay, *Deploying and Evaluating a Mixed Reality Mobile Treasure Hunt: Snap2Play*, Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'08)

Articles longs, conférences nationales avec comité de lecture

Coutrix, Nigay, *Interagir avec un objet mixte : Propriétés physiques et numériques*, Actes de la 19ème conférence en Interaction Homme Machine (IHM'07)

Coutrix, Nigay, Pasqualetti, Renevier, *RAZZLE : de la conception à l'évaluation d'un système mobile et multimodal*, Actes des Troisèmes Journées Francophones : Mobilité et Ubiquité 2006 (UBIMOB'06)

Coutrix, Nigay, Renevier, *Modèle d'Interaction Mixte : la Réalité Mixte à la Lumière des Modalités d'Interaction*, Actes des Deuxièmes Journées Francophones: Mobilité et Ubiquité 2005 (UBIMOB'05)

Références

Abou Moussa, Emmanuel Dubois, Christophe Bortolaso, Jean-Pierre Jessel, Cédric Bach. *SIMBA: Méthodologie et Plateforme de Prototypage Moyenne Fidélité pour les Systèmes Interactifs Mixtes.* Dans : *Ubiquité et Mobilité*, *Saint Malo*, *27/05/08-31/05/08*, E Dubois, JM Pierson (Eds.), ACM Press, p. 21-29, 2008.

Abowd, Coutaz, Nigay, 1992. *Structuring the Space of Interactive System Properties*, In IFIP TC2/WG2.7 Working Conference on Engineering for Human Computer Interaction, North Holland Publ., J. Larson &C. Unger Eds.

Allen, 1983. *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM 26(11), pp. 832 43.

Anranda, Ernst, Horkoff, Eastbrook, 2007. *A Framework for Empirical Evaluation of Model Comprehensibility*, Proceedings of the International Workshop on Modeling in Software Engineering at International Conference on Software Engineering.

Appert, 2007. *Modélisation, Évaluation et Génération de Techniques d'Interaction*. Thèse de doctorat de l'Université Paris-Sud. Mai 2007, Orsay, France.

Avrahami, Hudson, 2002. *Forming Interactivity: A Tool for Rapid Prototyping of Physical Interactive Products*, Proceedings of the ACM Symposium on Designing Interactive Systems, DIS'02, June 2002, pp. 141-146.

Bach, Scapin, 2005. *Critères Ergonomiques pour les Interactions Homme-Environnements Virtuels : définitions, justifications et exemples*, rapport de recherche INRIA n°5531, Mars 2005.

Bailly, Nigay, Auber, 2006. *NAVRNA : Viusalization – Exploration – Edition of RNA*. In proceedings of AVI'06, ACM Press, New York, NY, pp. 504-507.

Ballagas, Memon, Reiners, Borchers, 2007. *iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing*, Proceedings of the SIGCHI conference on Human factors in computing systems, CHI'07, San Jose, California, USA, p.1107-1116.

Ballagas, Ringel, Stone, Borchers, 2003. *iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments*, In Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA, April 2003, pp. 537-544.

Barkhuus, Chalmers, Hall, Tennent, Bell, Sherwood, Brown, 2005. *Picking Pockets on the Lawn: The Development of Tactics and Strategies in a Mobile Game,* Paper in Proc. Ubiquitous Computing (Ubicomp), Tokyo, LNCS 3660, pp. 358-374.

Bastien, Scapin, 1993. *Ergonomic criteria for the evaluation of human-computer interfaces*. (Technical report N° 156). Rocquencourt, France : INRIA Rocquencourt.

Beaudouin-Lafon, 2004. *Designing Interaction, not Interfaces*, Proceedings of the ACM AVI'04 Conference, pp. 15-22.

Beaudouin-Lafon, 2000. *Instrumental Interaction : An Interaction Model for Designing Post-WIMP User Interfaces*, In Proceedings of the ACM CHI 2000 Conference on Human Factors in Computing Systems, CHI Letters, pp. 446-453.

Bell, Chalmers, Barkhuus, Hall, Sherwood, Tennent, Brown, Rowland, Benford, Hampshire, Capra, 2006. *Interweaving Mobile Games With Everyday Life*, In Proceedings of the ACM CHI 2006 Conference on Human Factors in Computing Systems, CHI'06.

Benford, Fahlén, 1993. *A Spatial Model of Interaction in Large Virtual Environments*, In Proceedings of the ECSCW'93 Conference.

Benford, Schnädelbach, Koleva, Anastasi, Greenhalgh, Rodden, Green, Ghali, Pridmore, Gaver, Boucher, Walker, Pennington, Schmidt, Gellersen, Steed, 2005. *Expected, sensed, and desired: A framework for designing sensing-based interaction*, ACM Transactions on Computer-Human Interaction (TOCHI), Volume 12 , Issue 1 (March 2005), pp. 3-30.

Bernsen, 1993. *Taxonomy of HCI Systems : State of the Art*, ESPRIT BR GRACE, 1993, delivrable 2.1.

Blackwell, Green, 2000. *A Cognitive Dimensions Questionnaire*, http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDquestionnaire.pdf

Bolt, 1980, *Put-that-there: voice and gesture at the graphics interface*. SIGGRAPH'80, 14, 3, pp. 262-270.

Bouchet, Nigay, Ganille, 2004. *ICARE Software Components for Rapidly Developing Multimodal Interfaces*, In Proceedings of ICMI 2004, State College, Pennylvania, USA, pp. 251-258.

Bouchet, 2006. Thèse de doctorat, décembre 2006.

Brave, Ishii, Dahley, 1998. *Tangible interfaces for remote collaboration and communication*, In Proceedings of the 1998 ACM conference on Computer supported cooperative work, Seattle, Washington, United States, pp.169-178

Buxton, 1983. *Lexical and pragmatic considerations of input strcutures*, Computer Graphics, 17 (1), pp. 31-37.

Buxton, 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Editeur.

Campbell, 2001. http://www.jimcampbell.tv/

Card, Moran, Newell, 1983. *The psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

Carvey, Gouldstone, Vedurumudi, Whitone, Ishii, 2006. *Rubber Shark as User Interface*, In Proceedings of the Conference on Human Factors in Computing Systems, CHI'06, pp. 634-639.

Chalon, 2004. *Réalité mixte et Travail Collaboratif : IRVO, un modèle de l'interaction Homme-Machine*, Thèse de doctorat, décembre 2004.

Charfi, Scapin, Dubois, 2008. *Identification et prise en compte de propriétés ergonomiques pour la modélisation et la conception de SIM*, Actes de la conférence IHM 2008, Metz, ACM Press, pp. 55-62.

Cheung-Foo-Wo, Tigli, Lavirotte, Riveill, 2007. *Contextual Adaptation for Ubiquitous Computing Systems using Components and Aspect of Assembly*, in Applied Computing (IADIS), Salamanca, Spain, février 2007.

Chevallet, Lim, Leong, 2005. *Object Identification and Retrieval from Efficient Image Matching. Snap2Tell with the STOIC dataset*, in Asia Information Retrieval Symposium (AIRS 2005), Jeju Island, Korea, 13-15 October, 2005.

Chevallet, Lim, Leong, 2007. *Object Identification and Retrieval from Efficient Image Matching Snap2Tell with the STOIC dataset*, Information Processing & Management, vol43, no2, pp515-530, March 2007.

Chin, You, Coutrix, Lim, Chevallet, Nigay, 2008. *Mobile phone-based mixed reality: the Snap2Play game*, The Visual Computer, Springer Berlin / Heidelberg Publ., ISSN 0178-2789 (Print) 1432-2315 (Online), August 2008.

Chin, You, Coutrix, Lim, Chevallet, Nigay, 2008. *Snap2Play: A Mixed-Reality Game based on Scene Identification*, In Proceedings of the 14th International IEEE and ACM Multimedia Modeling Conference MMM 2008, Springer LNCS (Lecture Notes in Computer Science), Advances in Multimedia Modeling, Volume 4903/2008, Kyoto, Japan, January 9-11 2008, pp. 220-229.

Coutaz, Calvary, 2008. *HCI and Software Engineering: Designing for User Interface Plasticity*, In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications. pp. 1107-1125. 2008. Second Edition, ISBN 9780805858709, Taylor & Francis CRC Press, Human Factor and Ergonomics series, A. Sears, J. Jacko Eds. http://www.isrc.umbc.edu/HCIHandbook/

Coutrix, Nigay, Pasqualetti, Renevier, 2006. *RAZZLE : de la conception à l'évaluation d'un système mobile et multimodal*, Actes des Troisièmes Journées Francophones : Mobilité et Ubiquité 2006, UBIMOB 2006, Paris, France, 5-8 septembre 2006, ACM Press, pp. 1-8.

Dey, Mankoff, Abowd, Carter, 2002. *Distributed mediation of ambiguous context in aware environments*, Proceedings of the 15th annual ACM symposium on User interface software and technology, UIST'02, Paris, France, 2002, pp. 121-130.

Dix, Ghazali, Ramduny-Ellis, 2007. *Modelling Devices for Natural Interaction*. In Proceedings of FMIS 2007, 2nd International Workshop on Formal Methods for Interactive Systems, P. Curzon and A. Cerone (eds.), ENTCS, Vol 208C, pp. 23-40, Elsevier, 2007.

Dourish, 2004. *Where the Action is, The Foundations of Embodied Interaction*, MIT Press.

Dragicevic, Fekete, 2004. *Support for Input Adaptability in the ICON Toolkit*, In Proceedings of the ICMI 2004 Conference.

Dubois, Bach, Gauffre, Charfi, Salembier, 2006. *Instrumentation de Focus-Group dans la Co-Conception de Systèmes Mixtes*. Dans : Interaction Homme-Machine (IHM 2006), Montréal - Québec, 18/04/2006-21/04/2006, Jean-Marc Robert, Michel C. Desmarais, Eric Lecolinet, Bertrand David (Eds.), ACM Press, pp. 163-166, avril 2006

Dubois, Gray, 2007. *A Design-Oriented Information-Flow Refinement of the ASUR Interaction Model.* Dans : Engineering Interactive Systems (EHCI-HCSE-DSVIS 2007), Salamanca - Spain, 22/03/07-24/03/07, IFIP, p. 0-0, mars 2007.

Dubois, Nigay, Troccaz, Chavanon, Carrat, 1999. *Classification Space for Augmented Surgery, an Augmented Reality Case Study*, in conference proceedings of Interact'99 , Sasse,A., Johnson, C., (eds.), Edimburgh- UK, September 99, pp.353-359.

Dubois, Nigay, Troccaz, 2001. *Consistency in Augmented Reality Systems*, In Proceedings of the EHCI 2001 Conference, LNCS.

Dubois, 2001. *Chirurgie Augmentée, un cas de Réalité Augmentée. Conception et Réalisation centrée sur l'utilisateur*. Thèse de Doctorat, 2001.

Feiner, MacIntyre, Höllerer, Webster, 1997. *A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment*, ISWC 1997. (Int. Symp. on Wearable Computers), October 13-14, 1997, Cambridge, MA.

Fischer, Luge, Polk, 2008. *Carcade Game*, http://www.digital.udk-berlin.de/en/projects/winter0708/haupt/incargaming/carcade.html

Fishkin, 2004. *A taxonomy for and analysis of tangible interfaces*. Personal Ubiquitous Computing, Volume 8, Issue 5, Septembre 2004, pp. 347-358.

Fitzmaurice, Buxton, 1997. *An Empirical Evaluation of Graspable User Interfaces: towards specialized, space-multiplexed input*, In Proceedings of the CHI 1997 Conference, pp. 43-50.

Fitzmaurice, Ishii, Buxton, 1995. *Bricks :Laying the foundations for Graspable User Interfaces*, In Proceedings of the CHI Conference, pp.442-449.

Gaetano Adi, *Aexitimia*, 2006. http://www.paulagaetano.com.ar/, Alexitmia.

Gauffre, Dubois, Bastide, 2007. *Domain Specific Methods and Tools for the Design of Advanced Interactive Techniques*, Workshop MDDAUI'07.

Gaver, Martin, 2000. *Alternatives: exploring information appliances through conceptual design proposals*, in Proceedings of the SIGCHI conference on Human factors in computing systems, CHI'00, pp. 209-216.

Greenberg, Buxton, 2007. *Usability Evaluation Considered Harmful (Some of the Time)*. Report 2007-879-31, Department of Computer Science, University of Calgary, Calgary, AB, Canada. T2N 1N4. September.

Greenberg, Fitchett, 2001. *Phidgets: Easy Development of Physical Interfaces through Physical Widgets*. In Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology - ACM UIST'01. (Orlando, Florida) , November 11-14, ACM Press, pages 209-218.

Greenfield, 2006. *Everyware: the dawning age of ubiquitous computing*, ISBN 0321384016, 9780321384010, Published by New Riders.

Grudin, 1990. *The computer reaches out: the historical continuity of interface design*, Proceedings of the SIGCHI conference on Human factors in computing systems, Seattle, Washington, United States, ACM New York, NY, USA, pp. 261-268.

Harmann, Yu, Allison, Yang, Klemmer, 2008 (b). *Design as Exploration: Creating Interface Alternatives through Parallel Authoring and Runtime Tuning*, Proceedings of UIST'08, October 19 22, 2008, Monterey, California, USA, pp. 91-100

Hartmann, Doorley, Klemmer,, 2008. *Hacking, Mashing, Gluing: Understanding Opportunistic Design*. IEEE Pervasive Computing, July-September 2008.

Hartmann, Abdulla, Mittal, Klemmer, 2007. *Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition*, In proceedings of CHI'07, ACM Press, New York, NY, pp. 145-154.

Hartmann, Klemmer, Bernstein, Abdulla, Burr, Robinson-Mosher, Gee, 2006. *Reflective physical prototyping through integrated design, test, and analysis*, Proceedings of the 19th annual ACM symposium on User interface software and technology 2006, Montreux, Switzerland October 15 - 18, 2006 (UIST 2006), pp. 299-308.

Holmquist, Redström, Ljungstrand, 1999. Token-based access to digital information. In: Proceedings of the 1st international symposium on handheld and ubiquitous computing (HUC'99), Karlsruhe, Germany, September 1999, pp. 234 245.

Hudson, Mankoff, 2006. *Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape*, Proceedings of the ACM Symposium on User Interface Software and Technology, UIST'06, October 2006, pp. 289-297.

Iachello, 2006. *Privacy and Proportionality*, Thèse de Doctorat d'informatique, College of Computing, Interactive & Intelligent Computing Division, Georgia Institute of Technology, Mai 2006.

Ishii, Mazalek, Lee, 2001. *Bottles as a Minimal Interface to Access Digital Information*, In Proceedings of the Conference on Human Factors in Computing Systems, CHI 2001, Seattle, Washington, USA, March 31 - April 5, ACM Press.

Ishii, Ratti, Piper, Wang, Biderman, Ben-Joseph, 2004. *Bringing Clay and Sand into Digital Design - Continuous Tangible User Interfaces*, in BT Technology Journal, Vol. 22, No. 4, October 2004, pp. 287-299.

Ishii, Ullmer, 1997. *Tangible bits: towards seamless interfaces between people, bits and atoms*, Proceedings of the SIGCHI conference on Human factors in computing systems CHI'97, Atlanta, Georgia, United States, pp. 234-241.

Izadi, Hodges, Taylor, Rosenfeld, Villar, Butler, Westhues. 2008. *Going Beyond the Display: A Surface Technology with an Electronically Switchable Diffuser*. In Proc. ACM Symposium on User Interface Software and Technology (UIST '08). Monterey, CA, USA, 2008, pp. 269-278.

Jacob, Girouard, Hirshfield, Horn, Shaer, Solovey, Zigelbaum, 2008. *Reality-Based Interaction: A Framework for Post-WIMP Interfaces*, Proceedings of CHI 2008, April 5 10, 2008, Florence, Italy, ACM Press, pp. 201-210.

Jeremijenko, 1995. *Livewire*. http://www.nyu.edu/projects/xdesign/mainmenu/archive_livewire.html

Jeremijenko, 1995. *Voice Boxes*. http://www.nyu.edu/projects/xdesign/mainmenu/archive_voicebox.html

Johanson, Fox, Winograd, 2002. *The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms*, IEEE Pervasive Computing Magazine, 1(2), April June 2002.

Johnson, Gardner, Wiles, Sweetser, Hollingsworth, 2002, *The inherent appeal of physically controlled peripherals*, in Entertainment Computing: Technologies and Applications, eds R. Nakatsu & J. Hoshino, Kluwer Academic Publishers, USA, pp. 371 378

Jordà, Geiger, Alonso, Kaltenbrunner, 2007. *The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces*. In proceedings of TEI'07.

Kanis, Winters, Agamanolis, Gavin, Cullinan, 2005. *Toward Wearable Social Networking with iBand*, CHI 2005 Extended Abstracts on Human Factors in Computing Systems, Portland, Oregon, 2 - 7 April 2005, ACM Press.

Klemmer, Li, Lin, Landay, 2004. *Papier-Mâché: Toolkit Support for Tangible Input*. CHI Letters, Human Factors in Computing Systems: CHI 2004, 6(1).

Kultima, 2008. *Creativity Techniques in Game Design*, Proceedings of the Game Developers Conference Mobile.

Landay, 1996. *Interactive Sketching for the Early Stages of User Interface Design*, PhD thesis, CMU.

Lee, Avrahami, Hudson, Forlizzi, Dietz, Leigh, 2004. *The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices*, Proceedings of the ACM Symposium on Designing Interactive Systems, DIS'04, August 2004, pp. 167-175.

Lee, 2008. *Technology education for woman by d.i.y. technology in closing gender gap*, In CHI '08 extended abstracts on Human factors in computing systems, pp. 3447-3452.

Lim, Stolterman, Tenenberg, 2008. *The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas*, ACM Transactions on Computer-Human Interaction (TOCHI), Volume 15 , Issue 2 (July 2008), Article No. 7.

Mackay, Fayard, Frobert, Médini, 1998. *Reinventing the Familiar: Exploring an Augmented Reality Design Space for Air Traffic Control*. In Proceedings of ACM CHI '98 Human Factors in Computing Systems. Los Angeles, California: ACM/SIGCHI. ACM Press, pp. 558-565.

Mackay, 1996. *Augmented Reality : A new paradigm for interacting with computers*, La recherche.

Mackay, 1996. *Réalité Augmentée : le meilleur des deux mondes*. La Recherche, Special issue on L'ordinateur au doigt et à l'œil. Vol. 284., (March 1996).

Mackinlay, Card, Robertson, 1990. *A Semantic Analysis of the Design Space of Input Devices*, Human Computer Interaction, Lawrence Erlbaum Associates Inc. Hillsdale, NJ, USA, Volume 5, Issue 2 (&3), June 1990, pp. 145-190.

Mansoux, Nigay, Troccaz, 2006. *Output Multimodal Interaction: The Case of Augmented Surgery*, In Proceedings of HCI 2006, Human Computer Interaction, People and Computers, The 20th BCS HCI Group conference in co-operation with ACM (London, UK).

Marquardt, Greenberg, 2007. *Distributed Physical Interfaces with Shared Phidgets*. In Proc. 1st International Conference on Tangible and Embedded Interaction. (Baton Rouge, Louisiana, USA), ACM Press, February 15-17, pp. 13-20.

Milgram, Kishino, 1994. *A Taxonomy of Mixed Reality Visual Displays*, IEICE Transactions on Information Systems, Vol.E77-D, No.12., pp. 1321-1329.

Murray-Smith, Williamson, Hughes, Quaade, *Stane: Synthesized Surfaces for Tactile Input*, Proceedings of CHI '08 conference, pp.1299-1302.

Myers, Hudson, Pausch, 2000. *Past, Present and Future of User Interface Software Tools*, ACM Transactions on Computer Human Interaction, v7, n1, March 2000, pp. 3-28.

Nehls, http://www.hannes.200ok.de/

Nielsen, 1994. *Heuristic evaluation*. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY.

Nigay, Coutaz, Salber, Blanford, May, Young, 1995. *Four easy pieces for assessing the usability of multimodal interaction : the CARE Properties*, Proceedings of the INTERACT'95 conference.

Nigay, Coutaz, 1996. *Espaces conceptuels pour l'interaction multimédia et multimodale*, TSI spécial multimédia et collecticiel, AFCET et Hermes Publ., vol. 15(9), 1996, pp. 1195-1225.

Nigay, *Design Principles for Interactive Software*, 1997, ISBN : 0-412-72470-7, Gram, Cockton Directeurs de la publication, Chapman & Hall, Ltd. London, UK, Editeurs.

Nigay, Dubois, Renevier, Pasqualetti, Troccaz, 2003. *Mixed Systems : Combining physical and digital worlds*, HCI'03, Theory and Practice (Part I), Vol 1, Edited by Julie Jacko, Constantine Stephanidis, LEA, pp. 1203-1207.

Nigay, 1994. *Conception et Modélisation logicielles des systèmes interactifs : application aux interfaces multimodales*. Thèse de doctorat.

Norman, 1999. *Affordances, conventions and design*. Interactions 6 (3), pp. 38-43.

Norman, 1986. *Cognitive Engineering*, Book chapter of User Centered System Design, New Perspectives on Human-Computer Interaction, pp. 31-61.

Olsen, 2007. *Evaluating User Interface Systems Research*, Proceedings of the 20th annual ACM symposium on User interface software and technology, 2007, Newport, Rhode Island, USA, October 07 - 10, 2007 UIST 2007, ACM (2007), pp. 251-258.

Pangaro, Maynes-Aminzade, Ishii, 2002. *The Actuated Workbench: Computer-Controlled Actuation in Tabletop Tangible Interfaces*. In proceedings of UIST'02, ACM Press, New York, NY, pp. 181-190.

Patten, Ishii, 2007. *Mechanical Constraints as Computational Constraints in Tabletop Tangible Interfaces*. In proceedings of CHI'07, ACM Press, New York, NY, pp. 809-818.

Renevier, Nigay, 2001. *Mobile Collaborative Augmented Reality : the Augmented Stroll*, EHCI'01, IFIP WG2.7 (13.2) Working Conference, Toronto, Lecture Notes in Computer Science (LNCS), Vol. 2254, M. Reed Little, L. Nigay Eds, Springer-Verlag Publ., pp. 315-334.

Renevier, 2004. *Systèmes mixtes collaboratifs sur supports mobiles : Conception et Réalisation*, Thèse de Doctorat, juin 2004.

Rey, 2005. *Contexte en Interaction Homme-Machine : le contexteur*, Communication Langagière et Interaction Personne-Système - Fédération IMAG - Université Joseph Fourier - Grenoble I, Thèse préparée au sein de l'Université Joseph Fourier de Grenoble, 1er août, 2005, 186 pages.

Salber, Dey, Abowd, 1999. *The context toolkit: aiding the development of context-enabled applications*, Proceedings of the SIGCHI conference on Human factors in computing systems (CHI), Pittsburgh, Pennsylvania, United States, ACM New York, NY, USA, pp. 434-441.

Serrano, Juras, Nigay, 2008. *A Three-dimensional Characterization Space of Software Components for Rapidly Developing Multimodal Interfaces*, In proceedings of the 10th international conference on Multimodal interfaces, IMCI '08, October 20-22, Chania, Crete, Greece, pp. 149-156.

Shaer, Leland, Calvillo, Jacob, 2004. *The TAC Paradigm: Specifying Tangible User Interfaces*, Personal and Ubiquitous Computing, vol. 8, no. 5, Sept. 2004, pp. 359-369.

Shaw, 1994. *The Golden Calf*, http://www.jeffrey-shaw.net/html_main/show_work.php3?record_id=94

Shneiderman, 1997. *Direct manipulation for comprehensible, predictable and controllable user interfaces*, January 1997, Proceedings of the 2nd international conference on Intelligent user interfaces, IUI '97, pp. 33-39.

Shneiderman, 1983. *Direct Manipulation. A Step Beyond Programming Languages*, IEEE Transactions on Computers, Vol. 16, No. 8, August, pp. 57 69.

Terrenghi, 2008. *A Taxonomy for Exploring the Design Space of User-Display-Display Interactions*, Workshop on designing multi-touch interaction techniques for coupled public and private displays (PPD'08), AVI'08.

Terrenghi, Kirk, Richter, Krämer, Hilliges, Butz, 2008. *Physical handles at the interactive surface: exploring tangibility and its benefits*, Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy, pp. 138-145.

Thomas, Close, Donoghue, Squires, De Bondi, Morris, Piekarski, 2000. *ARQuake: An Outdoor/Indoor Augmented Reality First Person Application*. In Proceedings of Fourth International Symposium on Wearable Computers (ISWC 2000), IEEE Computer Society, Atlanta, October 2000, pp. 139-146.

Truong, 2005. *INCA: An Infrastructure for Capture & Access Supporting the Generation, Preservation and Use of Memories from Everyday Life*, PhD Thesis, august 2005.

Ullmer, Ishii, 2001. *Emerging Frameworks for Tangible User Interfaces*. In HCI in the New Millenium, John M. Carroll, Ed., pp. 579 601.

Ullmer, Ishii, Jacob, 2005. *Token+constraint systems for tangible interaction with digital information*, ACM Transactions on Computer-Human Interaction (TOCHI), Volume 12 , Issue 1, (March 2005), pp. 81-118.

Ullmer, Ishii, 1997. *The metaDESK: Models and Prototypes for Tangible User Interfaces*. In Proceedings of UIST'97, ACM Press, New York, NY, pp. 223-232.

Underkoffler, Ishii, 1999. *Urp: a luminous-tangible workbench for urban planning and design*. In: Proceedings of the CHI'99 conference on human factors in computing systems, Pittsburgh, Pennsylvania, May 1999, pp 386 393.

Vernier, Nigay, 2000. *A framework for the combination and characterization of output modalities*. In Proceedings of DSV-IS, pp. 32-48.

Weiser, 1994. *Building Invisible Interfaces*. Invited talk, Proc. ACM UIST '94.

Wellner, 1993. *Interacting with paper on the DigitalDesk*, Communications of the ACM, 36, 7 (July 1993), pp. 87-96.

Williamson, Murray-Smith, Hughes, 2007. *Shoogle: Multimodal Excitatory Interfaces on Mobile Devices*, In Proceedings of the CHI 2007 Conference.

You, Chin, Lim, Chevallet, Coutrix, Nigay, 2008. *Deploying and Evaluating a Mixed Reality Mobile Treasure Hunt: Snap2Play*, In Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI 2008, Amsterdam, the Netherlands, September 2-5 2008, ACM Press, pp. 335-338.

Annexe A :
Modélisation des exemples

1. DigitalDesk (bouton de papier FILL)



(a) (b) (c)

Figure 266: Exemple d'interaction avec le DigitalDesk : l'utilisateur place (a) et appuie sur (b) le bouton FILL, pour remplir le dessin du toit de la maison par des tuiles (c).

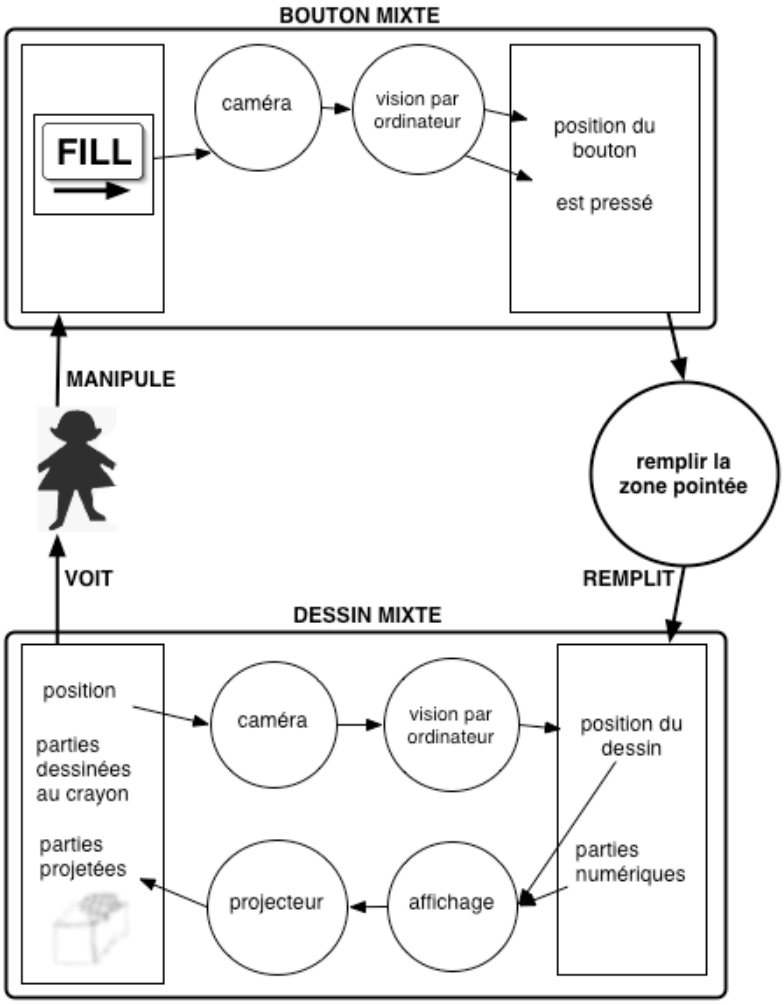


Figure 267: Description d'une interaction avec le DigitalDesk : l'utilisateur place et appuie sur le bouton FILL, pour remplir le dessin du toit de la maison par des tuiles.

2. SandScape

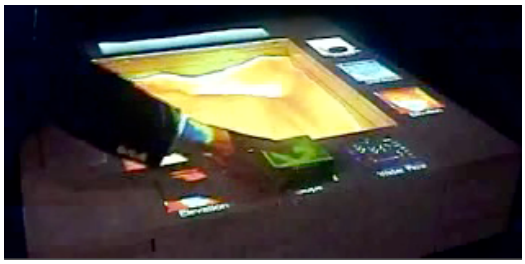


Figure 268: Exemple d'interaction avec Sandscape : l'utilisateur place le marqueur sur un emplacement prévu, pour choisir une simulation.

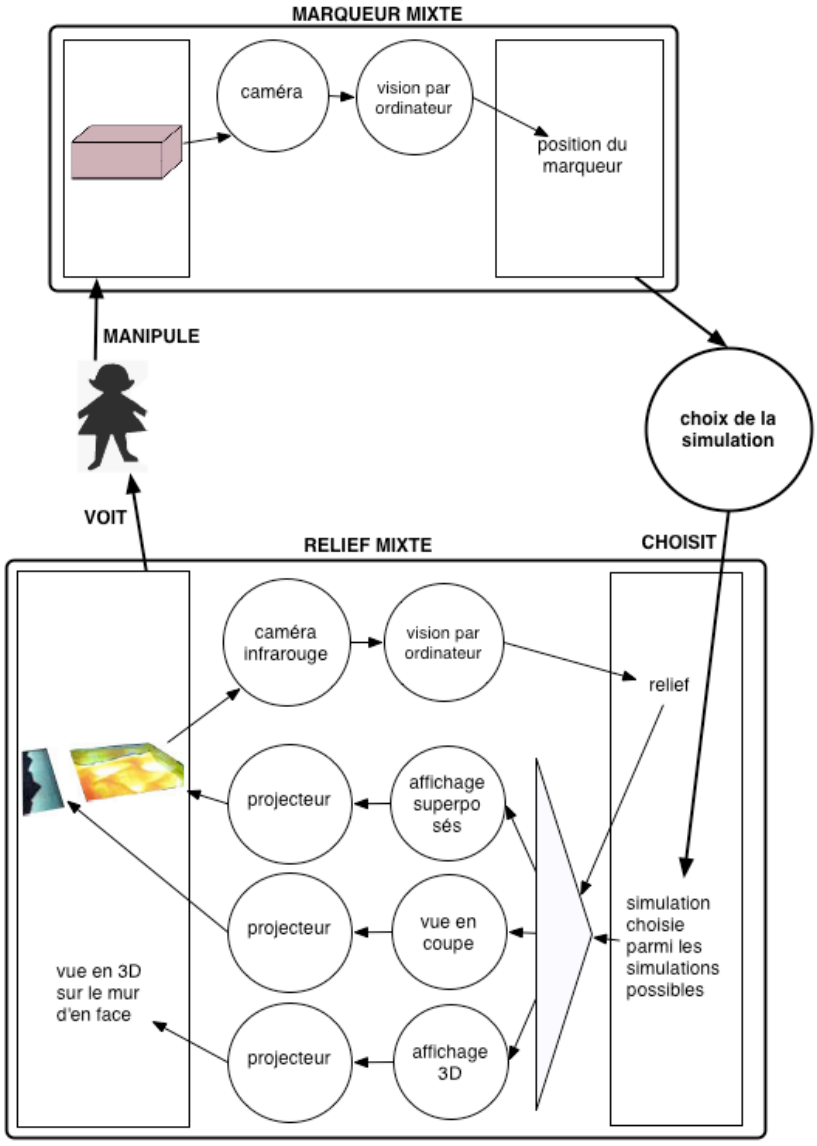


Figure 269: Description de l'interaction avec Sandscape.

3. Actuated workbench

3.1. Manipulation avec la trackball

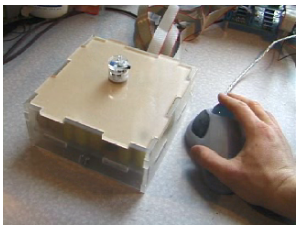


Figure 270: Interaction avec le palet de l'*actuated workbench*, à l'aide d'une trackball.

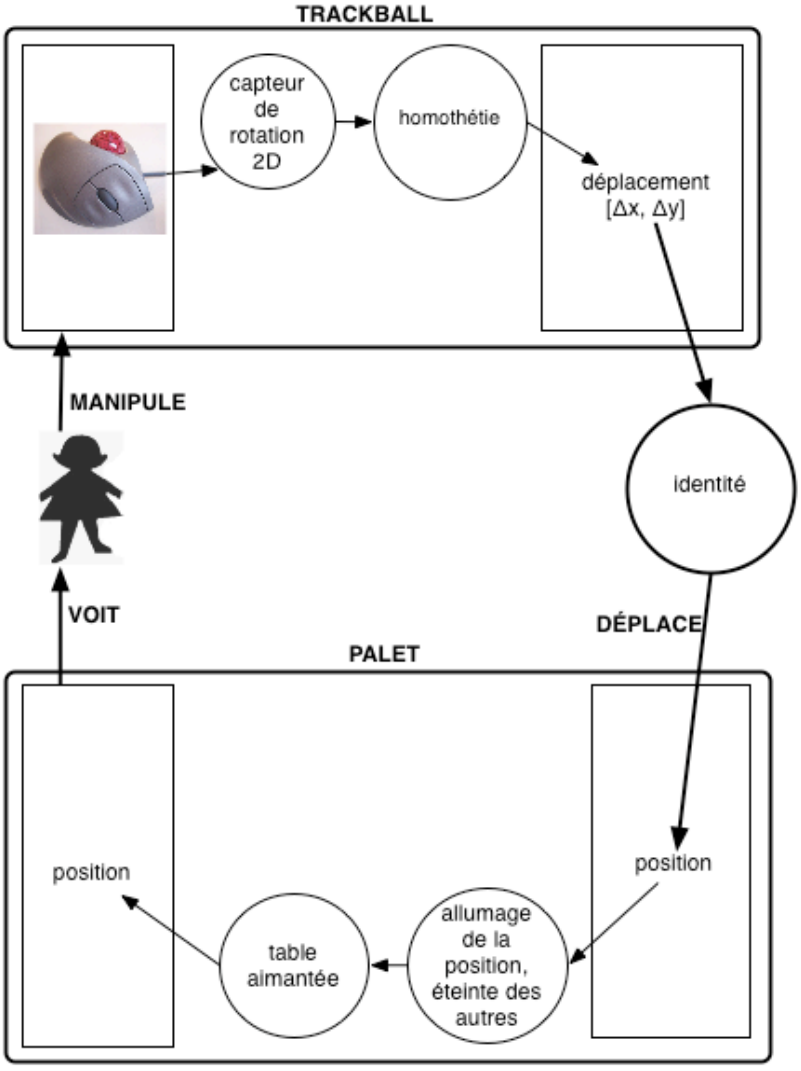


Figure 271: Description de l'interaction avec l'*actuated workbench*, à l'aide d'une trackball.

3.2. Manipulation directe

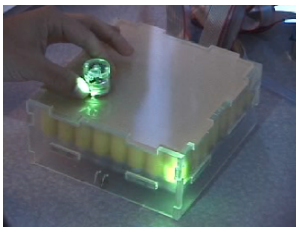


Figure 272: Interaction avec le palet de l'*actuated workbench* en mode vision par ordinateur.

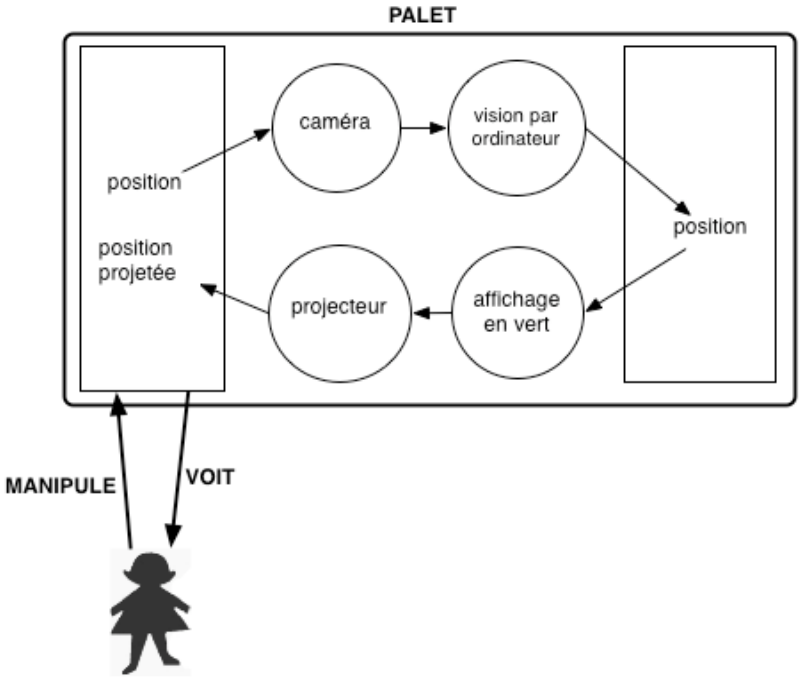


Figure 273: Description de l'interaction avec l'*actuated workbench*, en mode vision par ordinateur. Il n'y a pas de tâche, donc de notions d'outil ou objet de la tâche dans ce scénario. Cet objet mixte pourrait être l'un ou l'autre, selon l'application.

4. PICO

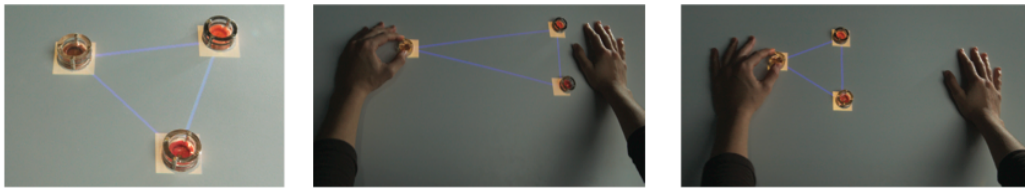


Figure 274 : Le système PICO (illustration issue de http://www.jamespatten.com/pico/picovid.html).

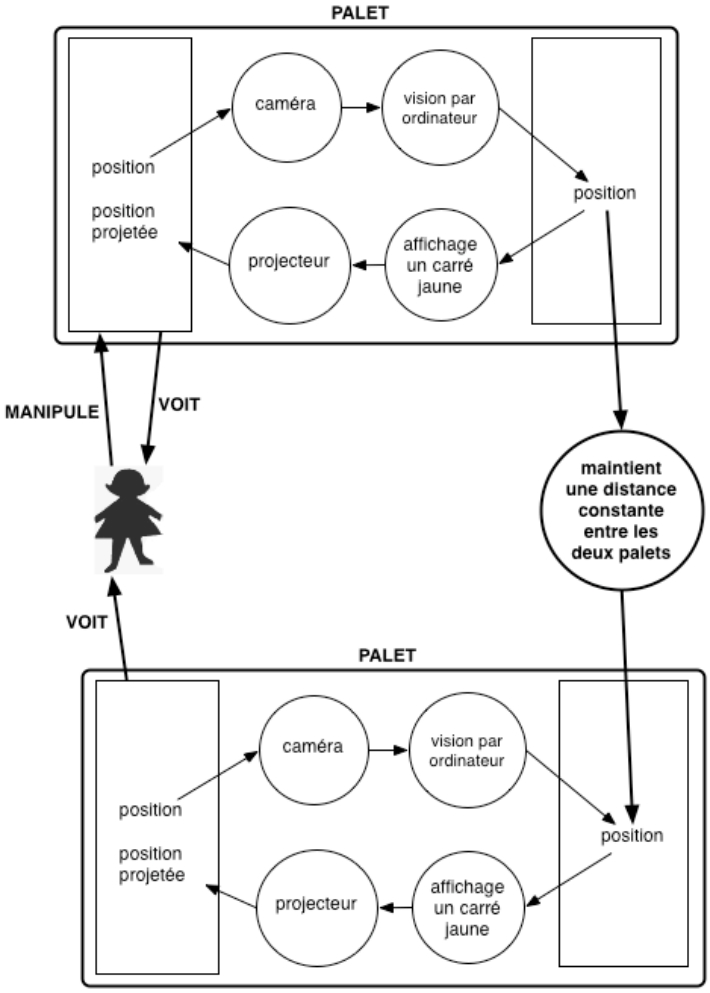


Figure 275 : Description de l'interaction avec deux palets de PICO.

5. CASPER, première version



Figure 276: Interaction avec CASPER (Computer ASsisted PERicardial surgery).

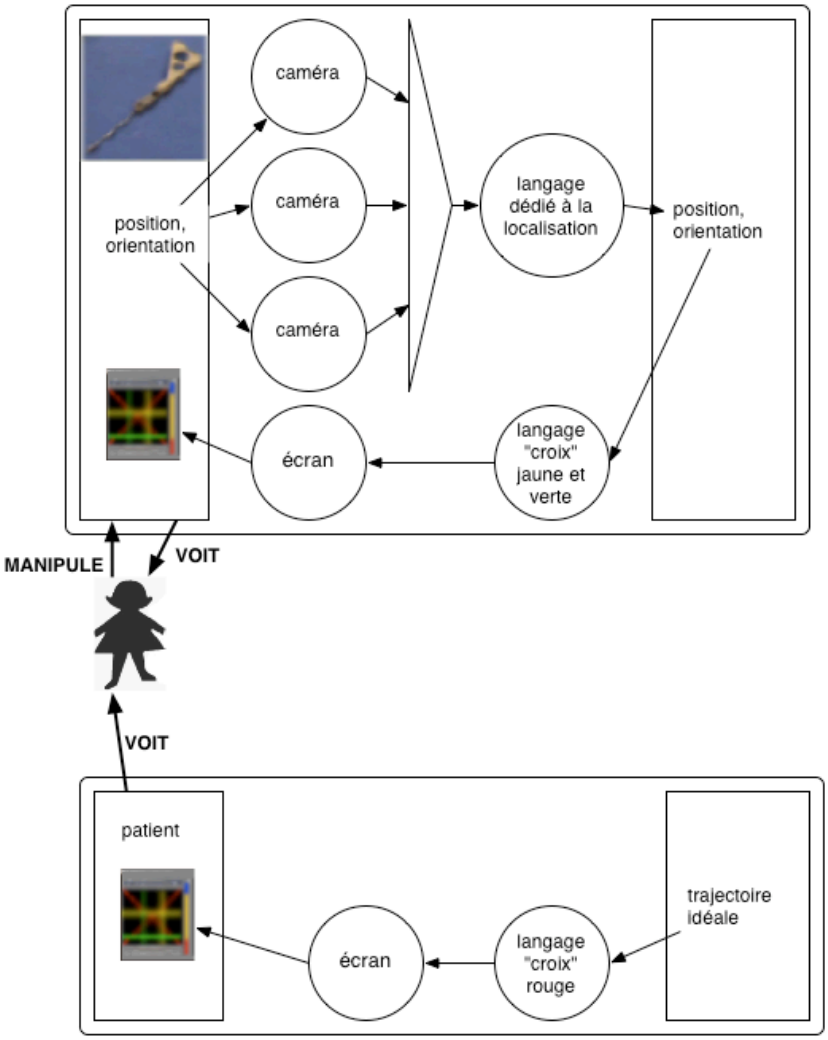
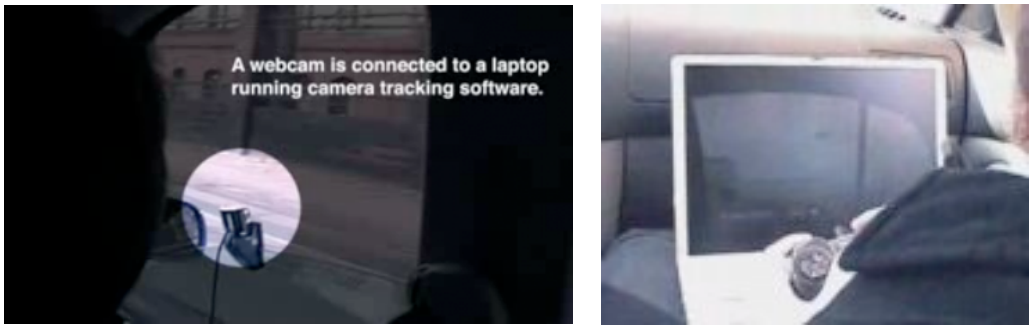


Figure 277: Description de l'interaction avec CASPER, première version. Il n'y a pas ici de langage d'interaction entre les deux objets : il s'agit d'un système de réalité augmente où seule l'évaluation est augmentée.

6. Carcade



(a) (b)

Figure 278 : Installation de Carcade dans une voiture : un camera sur la vitre capte l'image du décor qui défile (a), et l'utilisateur manipule l'avion grâce à une manette de jeu et voit l'avion et le paysage sur son écran (b).

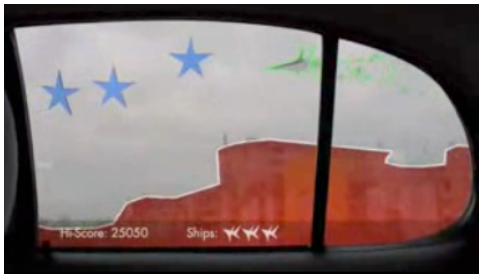


Figure 279: Interaction avec Carcade : des éléments numériques, comme les étoiles bleues, sont ajoutés au décor.



(**a**)



(**b**)

Figure 280: Interaction avec Carcade : Un élément du décor physique (immeuble) est reconnu et barre le chemin de l'avion. Le système ajoute un passage numérique (a) (en vert). Si le joueur ne présente pas à temps son avion en face du passage, l'avion s'écrase (b).

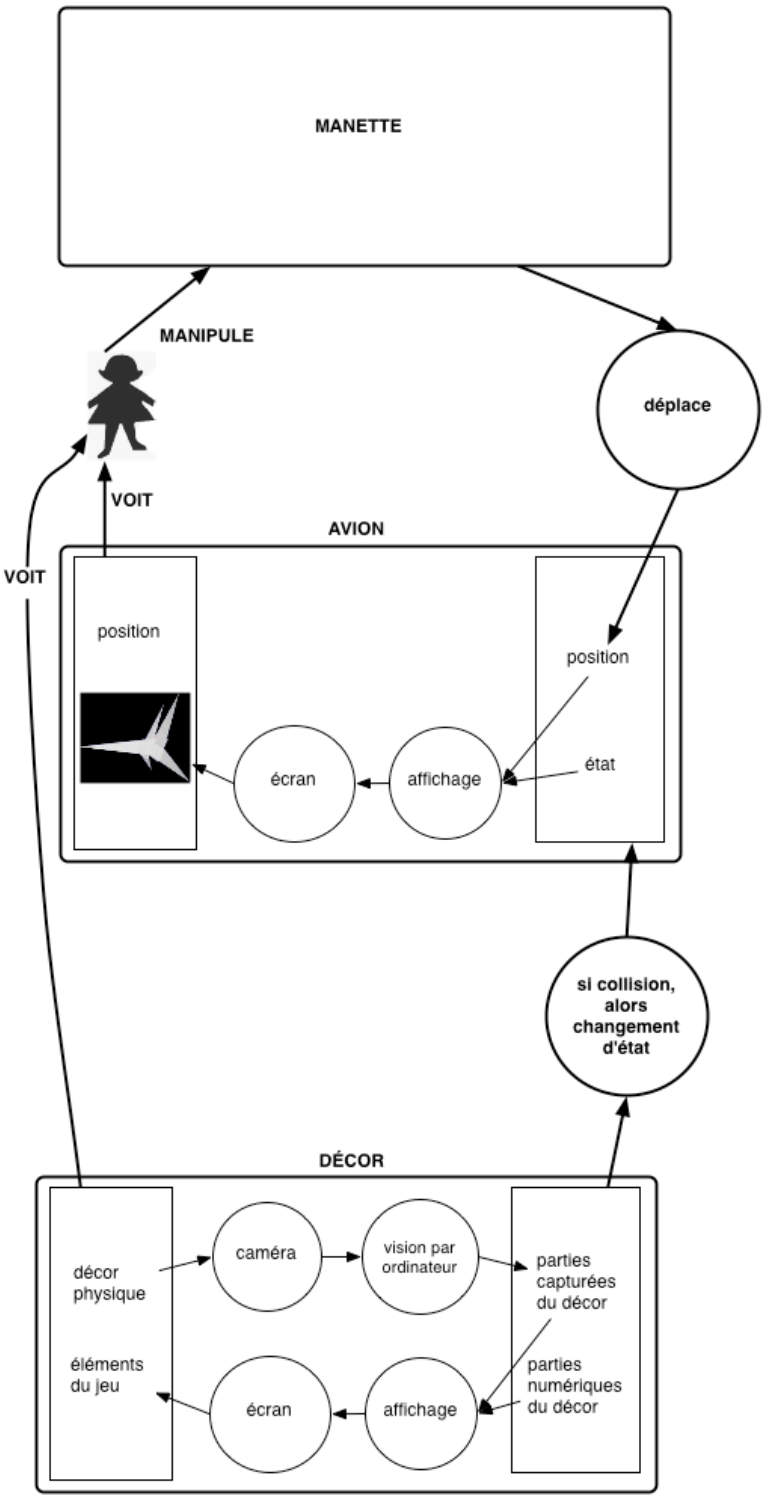


Figure 281: Description de l'interaction avec Carcade : La manette n'est pas détaillée faute d'informations.

7. The Golden Calf

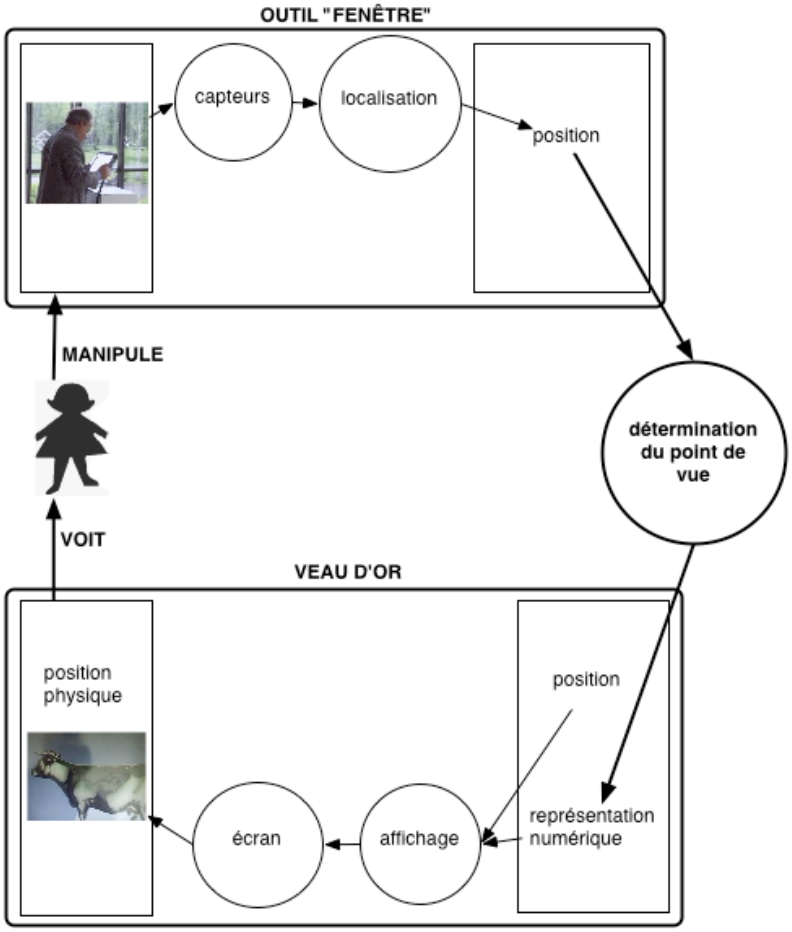


Figure 282: Description de l'interaction avec l'installation interactive *The Golden Calf* [Shaw, 1994].

8. Alexitimia



Figure 283: *Alexitimia*, Paula Gaetano, 2006 (d'après [Gaetano Adi, 2006]).

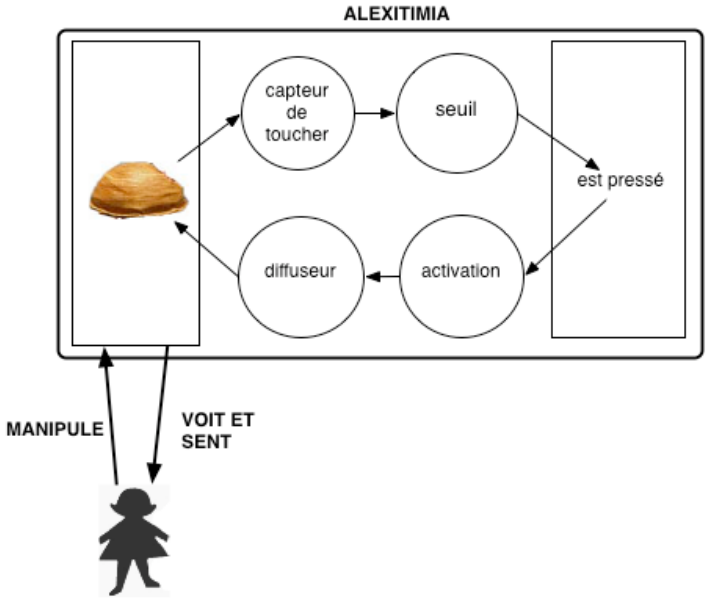


Figure 284 : Description de l'interaction avec Alexitimia [Gaetano Adi, 2006].

9. Voice Boxes



Figure 285: *Voice Boxes*, Natalie Jaremijenko, 1995 (d'après [Jeremijenko, 1995]).

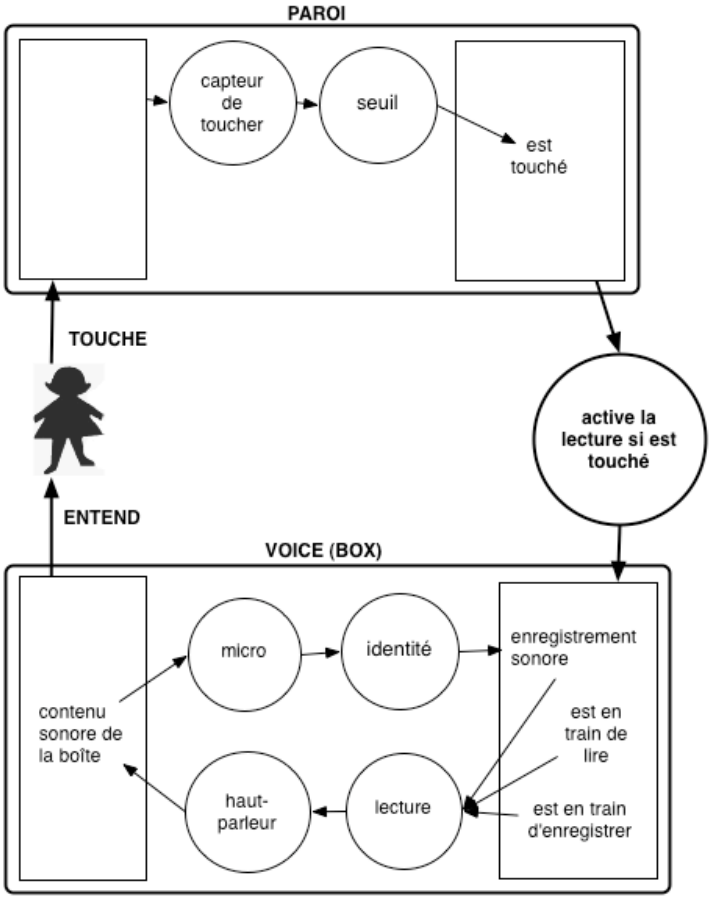


Figure 286 : Description de l'interaction pour la lecture du contenu sonore de la *voice box* [Jeremijenko, 1995].

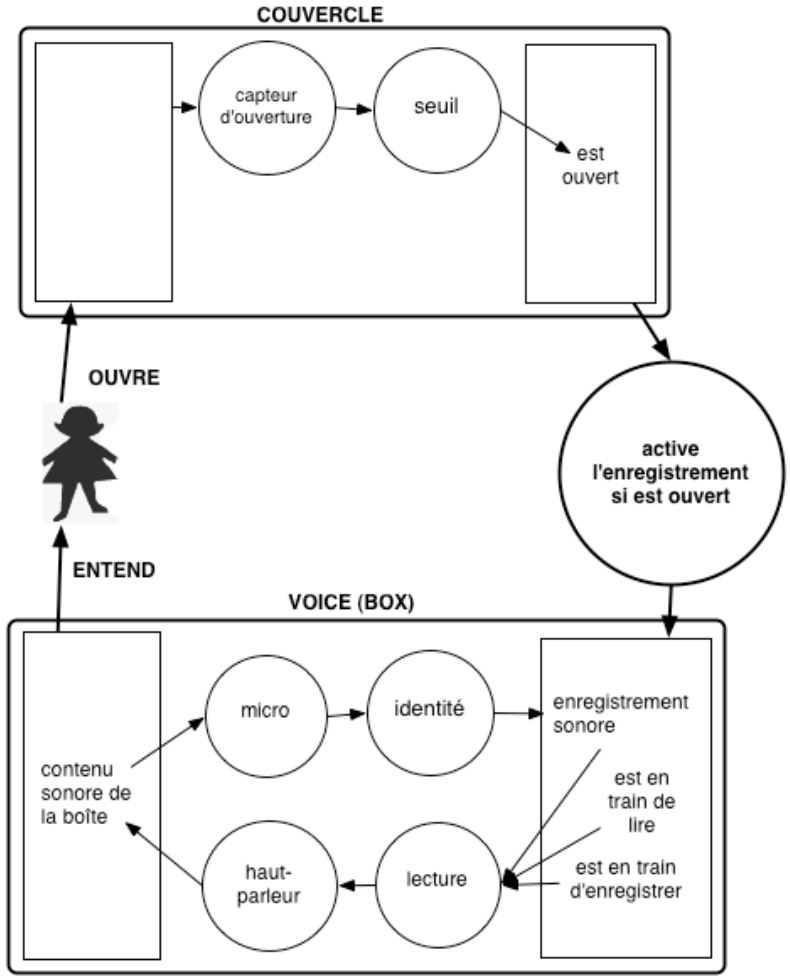


Figure 287 : Description de l'interaction pour l'enregistrement du contenu sonore de la *voice box* [Jeremijenko, 1995].

10. LiveWire



Figure 288: *LiveWire*, Natalie Jeremijenko, 1995 (d'après [Jeremijenko, 1995]).

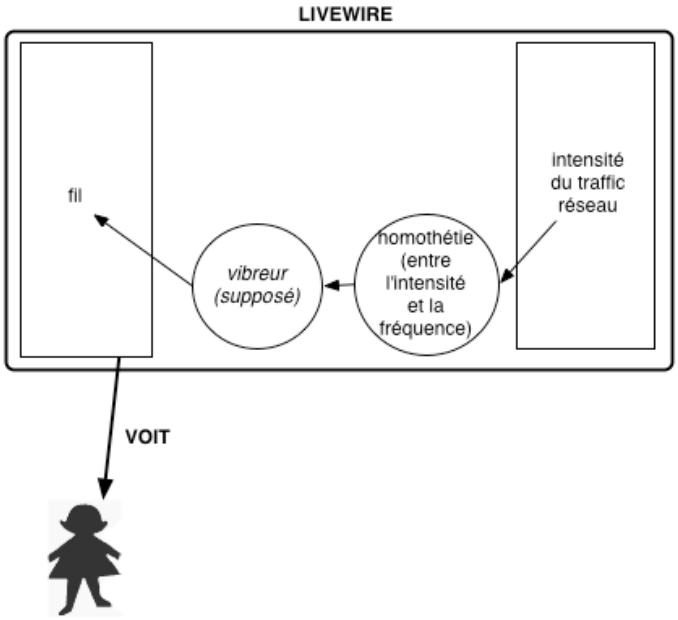


Figure 289 :Description de l'interaction avec le *LiveWire* [Jeremijenko, 1995].

11. The Eye Of Judgment



Figure 290: The Eye of Judgment pour PS3 (d'après http://www.eyeofjudgment.com/).

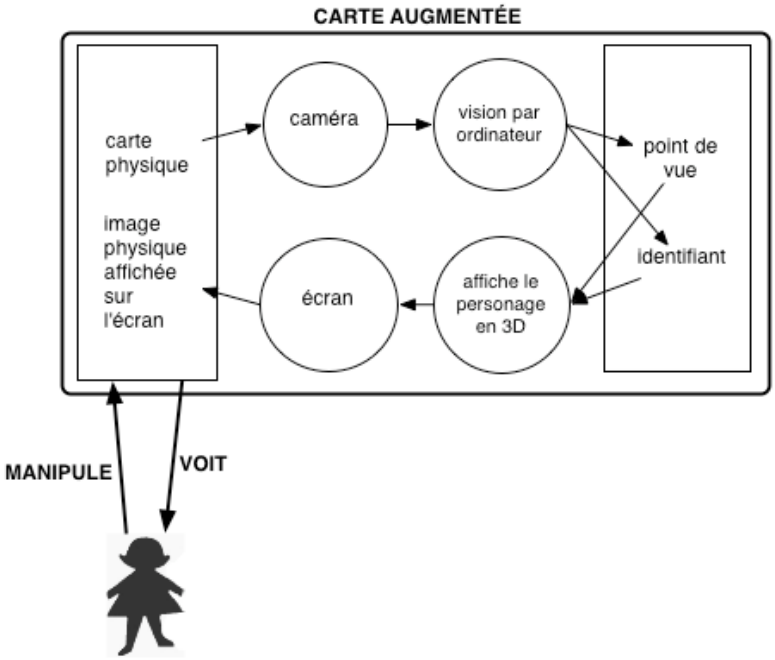


Figure 291 : Description d'une interaction avec une carte de The Eye of Judgment. Il n'y pas de langage d'interaction car il s'agit ici de réalité augmentée où seule l'évaluation de l'objet est augmentée.

12. reacTable

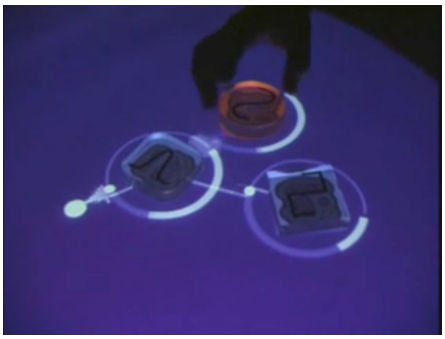


Figure 292: Un outil de la reacTable dont une propriété est contrôlée par un autre outil : Le premier est un filtre sonore passe-bas (objet bleuté aux coins arrondis avec une forme d'onde en pic représentée dessus). Le second est un oscillateur à basse fréquence (objet circulaire et rouge avec une sinusoïde représentée dessus).

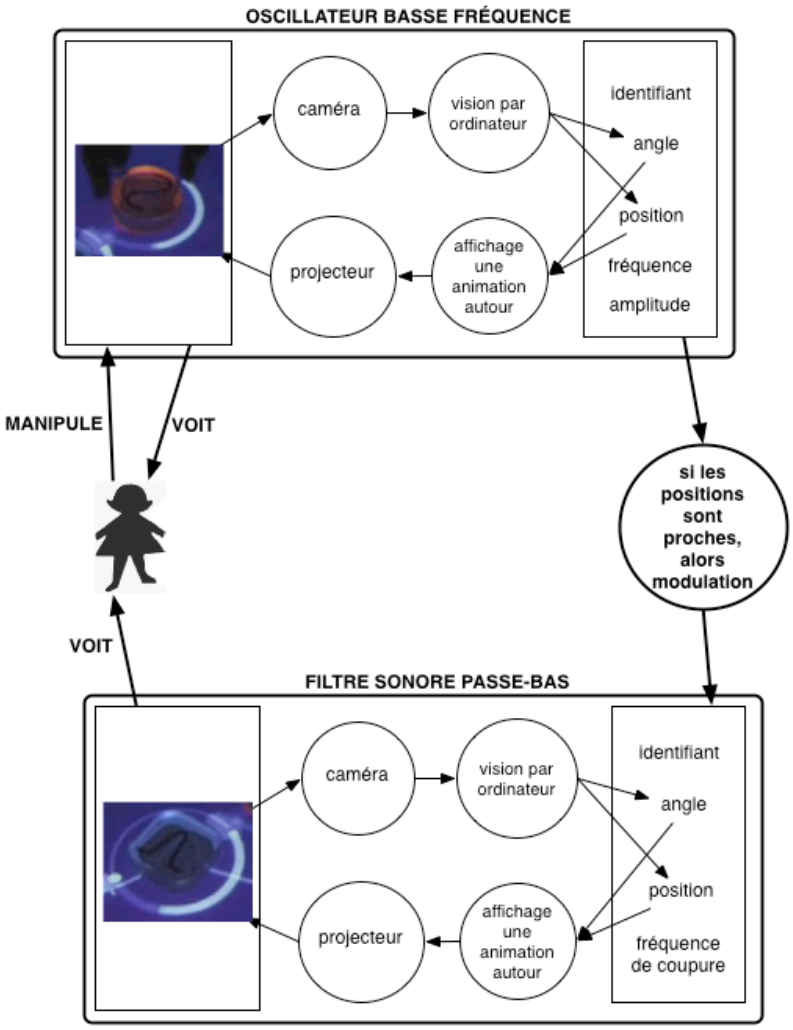


Figure 293 : Description d'une interaction avec la reacTable.

13. ARQuake



Figure 294 : Une vue dans le casque semi-transparent d'un joueur de l'ARQuake [Thomas et al., 2000] : on y voit à la fois le monde physique et les éléments ajoutés par l'ordinateur.

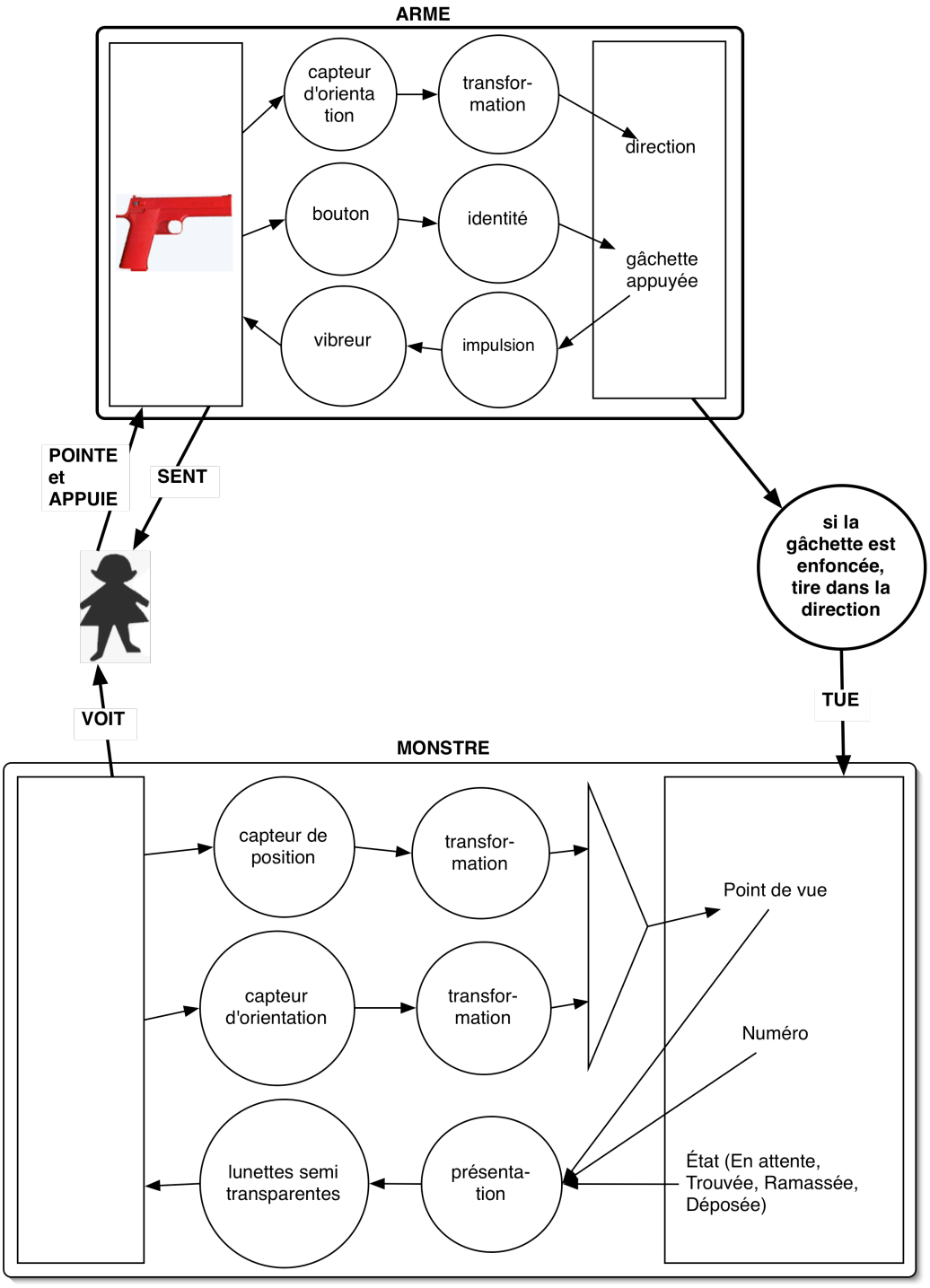


Figure 295 : Description d'une interaction avec ARQuake [Thomas et al., 2000].

14. Caméléon

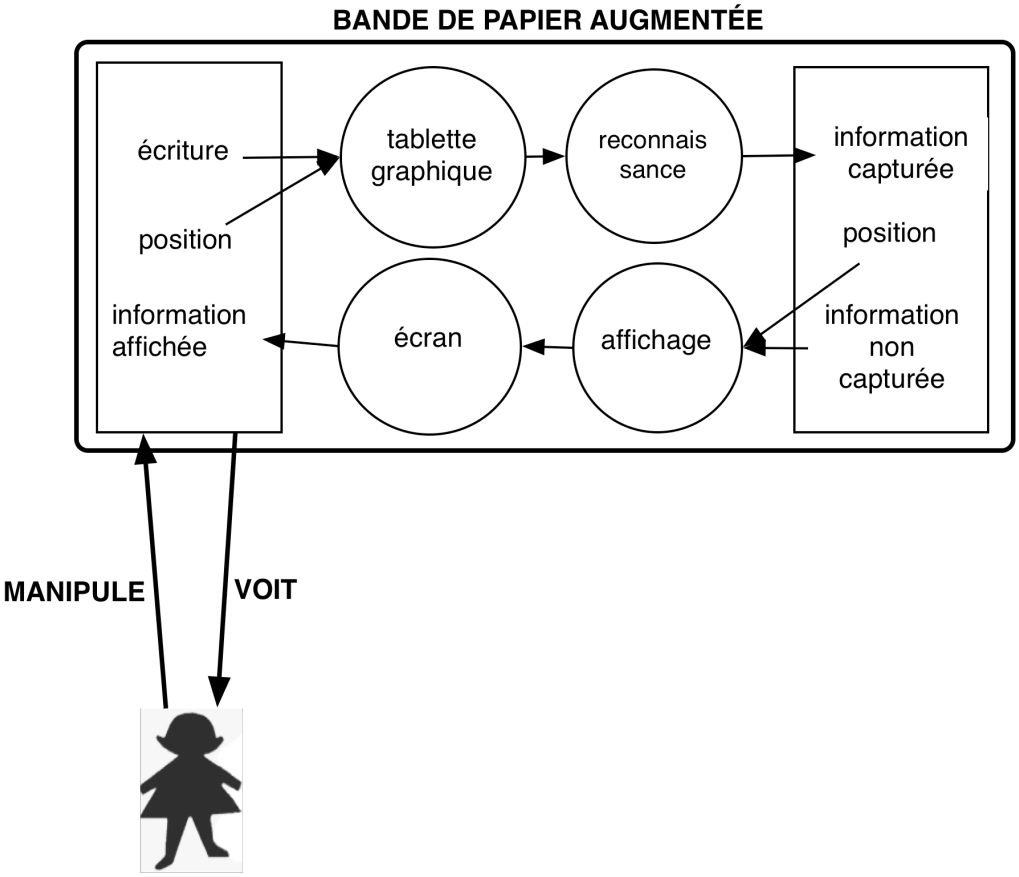


Figure 296 : Description d'une solution de conception pour les bandes de papier augmentées du système pour les contrôleurs aériens [Mackay et al., 1998] : ce qui est écrit sur la bande de papier sa position sont capturés avec une tablette graphique. L'information associée à la bande de papier est affichée sur un écran.

Annexe B : Solutions conçues par le focus group

1. Solutions générées pour l'objet de la tâche

Les idées générées lors de la séance et correspondant à l'objet de la tâche sont représentées dans la **Figure 297** et la **Figure 298**. Le groupe s'est mis d'accord pour interagir avec l'arbre de classification, qui est un objet d'origine numérique. **Figure 297** sont présentées les différentes possibilités d'augmentation par une partie physique qui ont été évoquées :

- Générer des propriétés physiques via des modalités de liaison en sortie,
- Ajouter à ces propriétés physiques générées une partie physique non générée par la machine : une représentation creuse de l'arbre, dans laquelle un outil de déplacement pourra circuler.

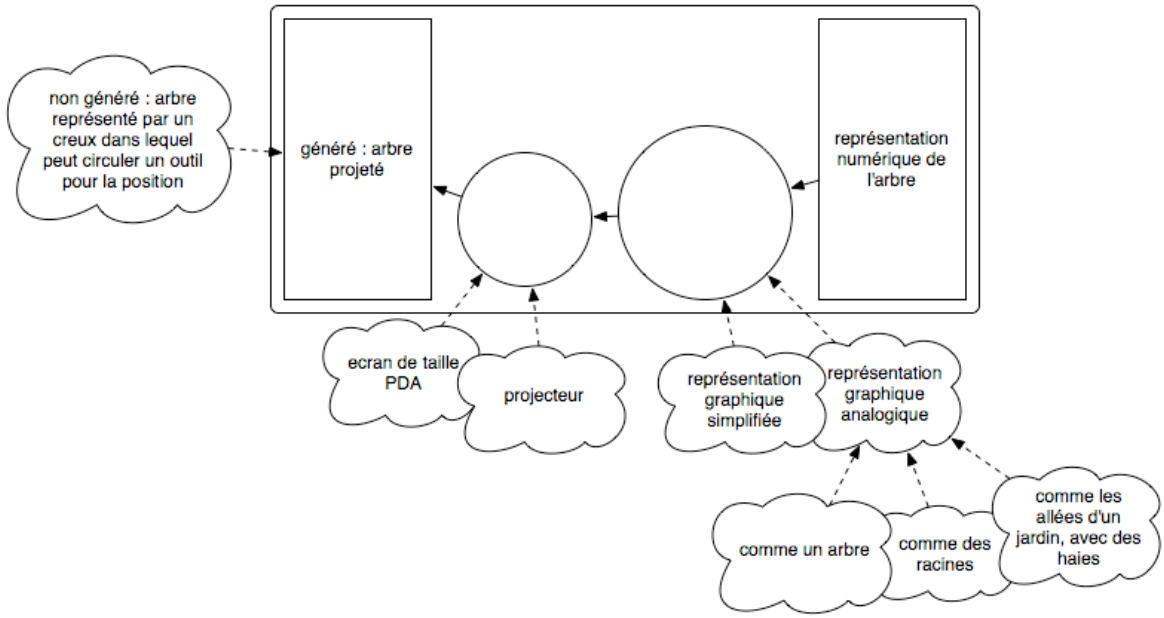


Figure 297 : Premières idées générées pour l'objet de la tâche.

Une autre solution pour l'objet de la tâche a été construite : La racine de l'arbre construit est représentée physiquement par une tête d'un animal (comme un cervidé par exemple), et les différentes branches explorées sont ses cornes. La **Figure 298** illustre cette solution. Pour cette solution, on remarque que la question de la composition spatiale entre les propriétés physiques générées et non-générées a clairement été discutée.

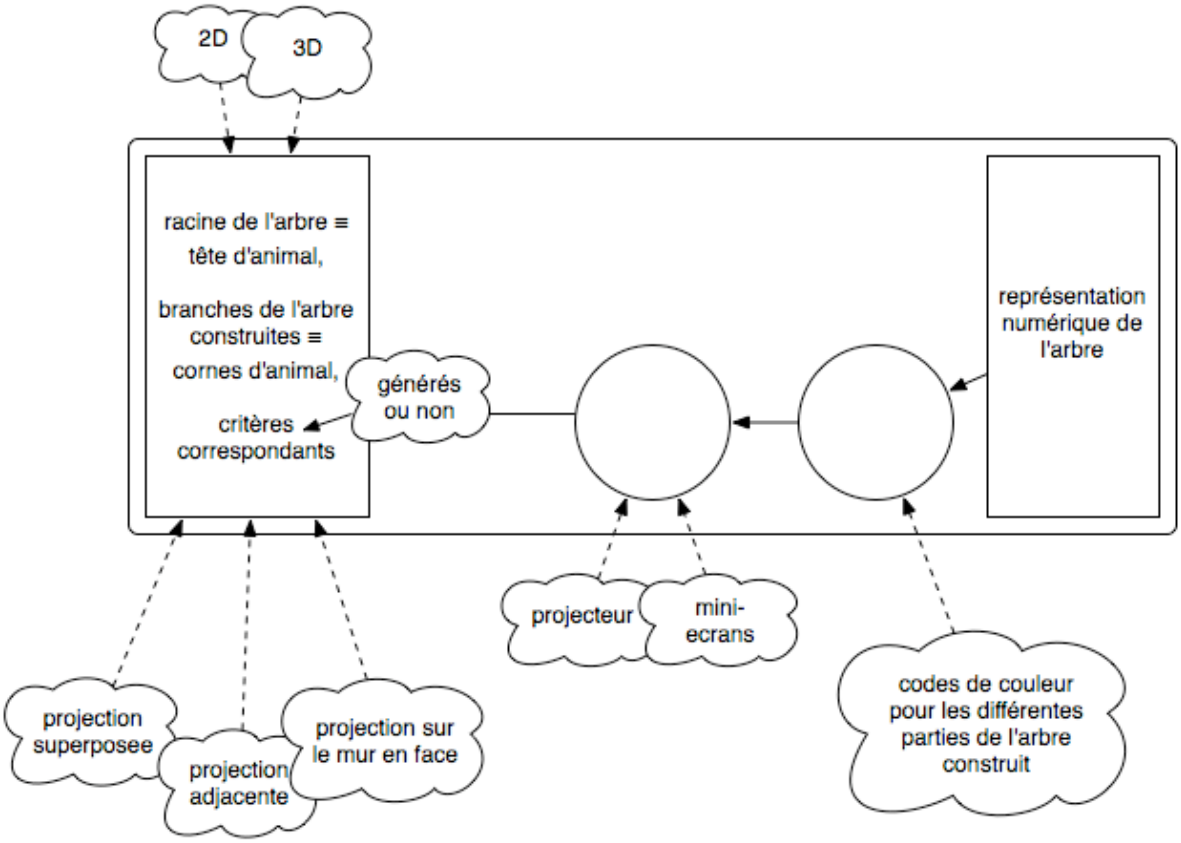


Figure 298 : Objet de la tâche sous forme de cornes d'animal.

2. Solutions générées pour l'interaction

La réflexion sur les outils pour les tâches « Avancer/reculer d'un niveau dans l'arbre » et « Se déplacer sur un noeud frère adjacent » a donné les solutions suivantes :

Pour le premier objet de la tâche (**Figure 297**) :

- Une barre d'onglets faisant l'analogie avec les onglets des intercalaires d'un classeur, pour avancer, reculer et voir les frères adjacents (de la **Figure 299** à la **Figure 304**),
- Un outil « levier de vitesse » pour avancer, reculer et voir les frères adjacents (**Figure 305**),
- Un outil à poser pour avancer ou reculer. Pour cet outil, différentes formes ont été imaginées, comme le montre la **Figure 306**.

Pour le second objet de la tâche (**Figure 298**) :

- Une partie élémentaire des cornes de l'animal peut servir d'outil d'interaction pour construire l'arbre, objet de la tâche (**Figure 307**).

Les solutions pour la plupart ont été partiellement conçues, ce qui explique les parties absentes dans les schémas.

2.1. Barre d'onglets

Les schémas de la **Figure 299** et **Figure 300**, **Figure 301**, **Figure 303** et **Figure 304** illustrent les solutions envisagées avec la barre d'onglets pour chaque tâche « avancer », « reculer » et « se déplacer vers un critère frère ». On remarque qu'elles proposent à la fois une métaphore de nom et de verbe.

Pour ces solutions, la question de la composition spatiale entre les outils pour avancer, reculer, et l'objet de la tâche a été étudiée. Une illustration de cette proposition de composition spatiale est schématisée **Figure 302**.

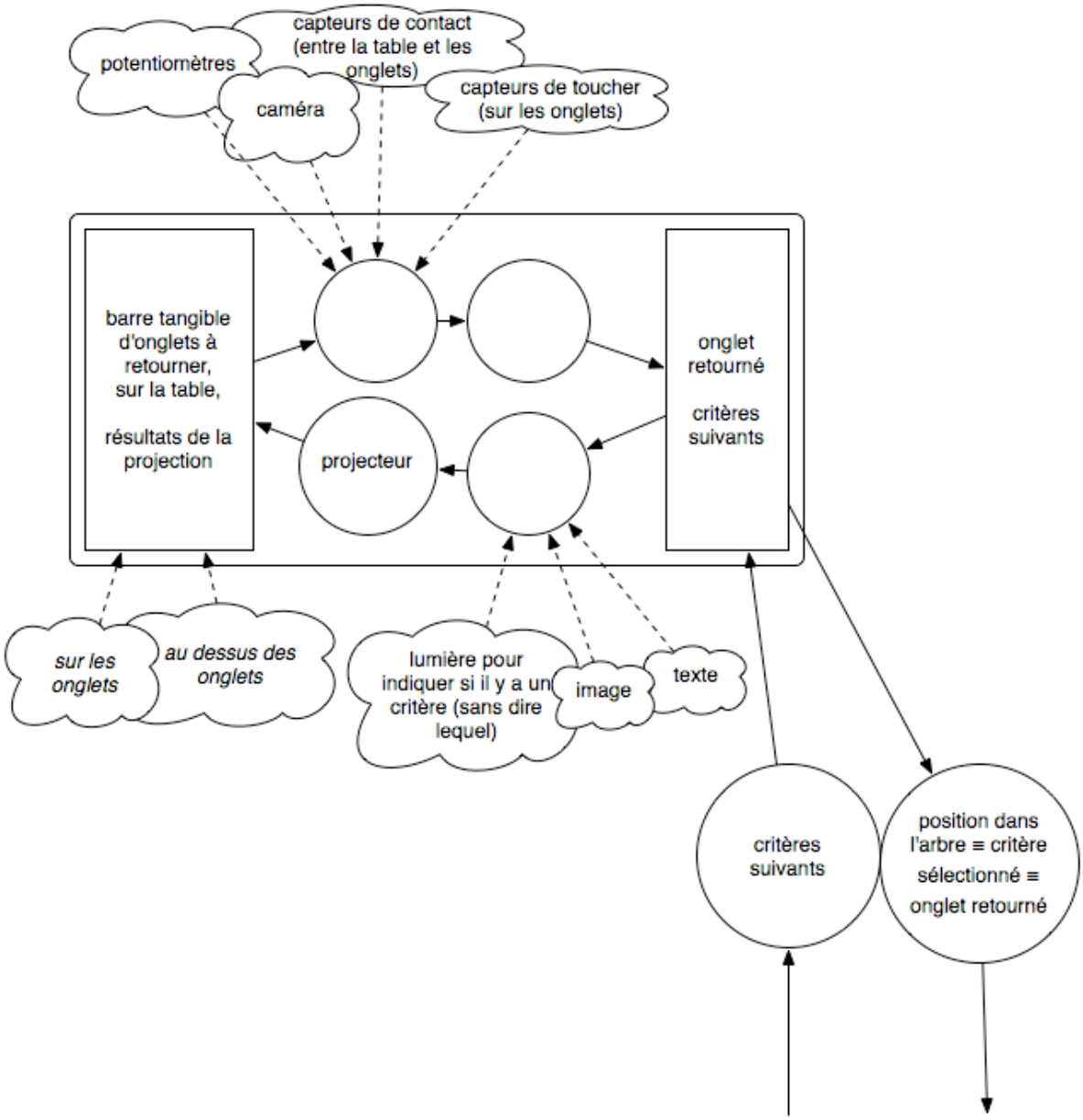


Figure 299 : Une barre d'onglets pour avancer dans l'arbre.

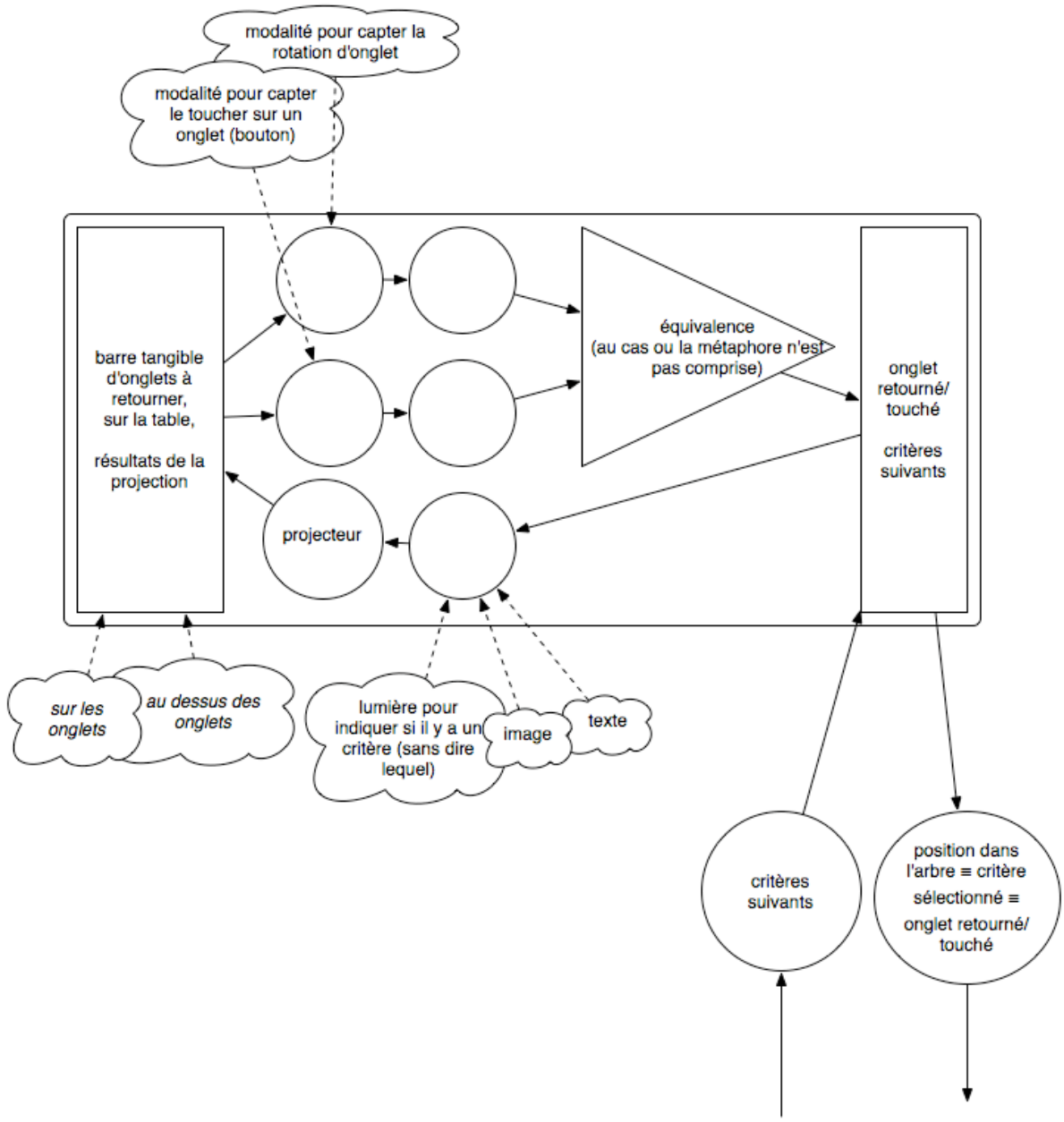


Figure 300 : Variante de la barre d'onglet pour avancer dans l'arbre : deux modalités équivalentes sont proposées en entrée, au cas où la métaphore ne serait pas comprise.

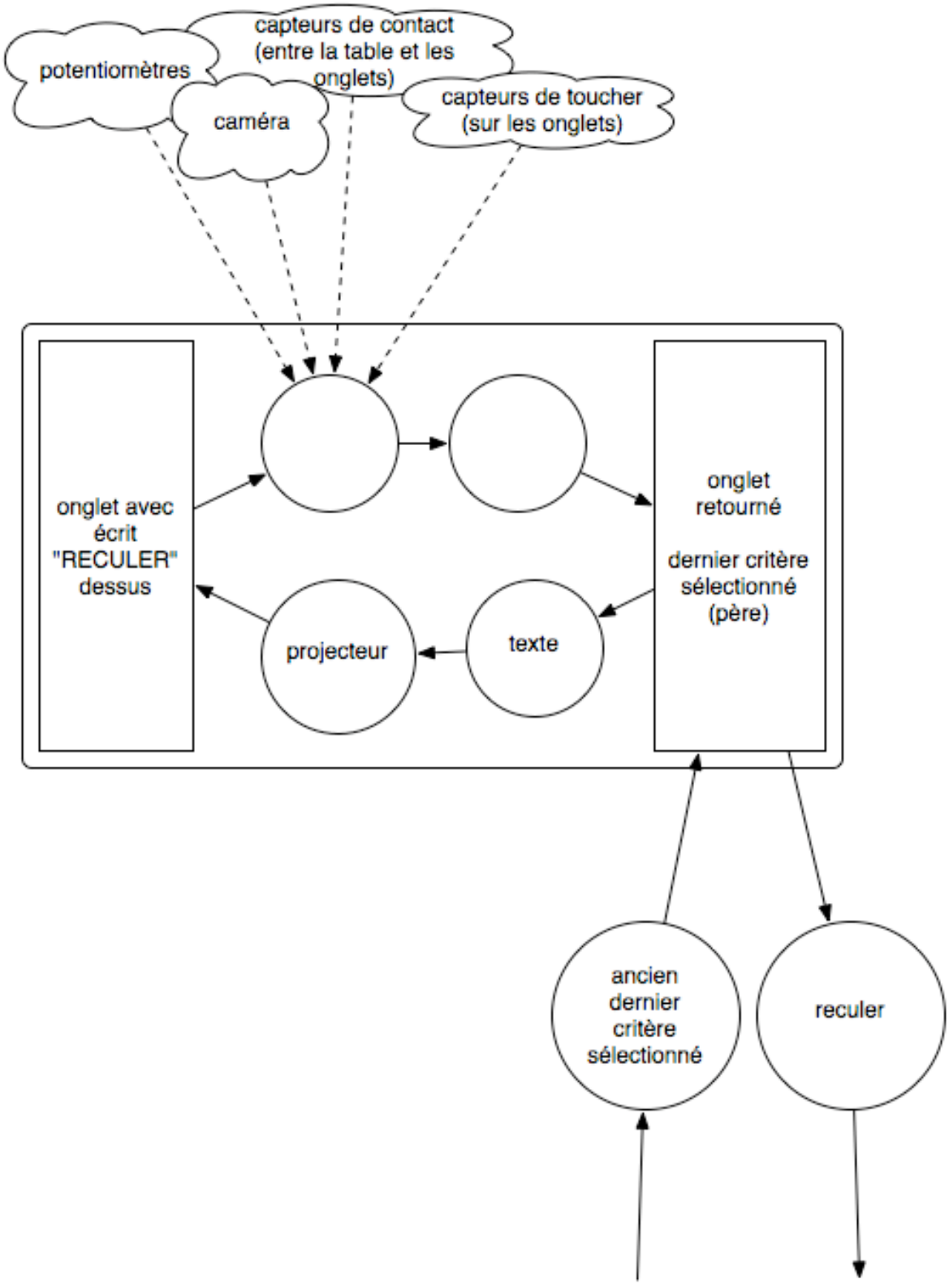


Figure 301 : Onglet pour reculer d'un niveau dans l'arbre.

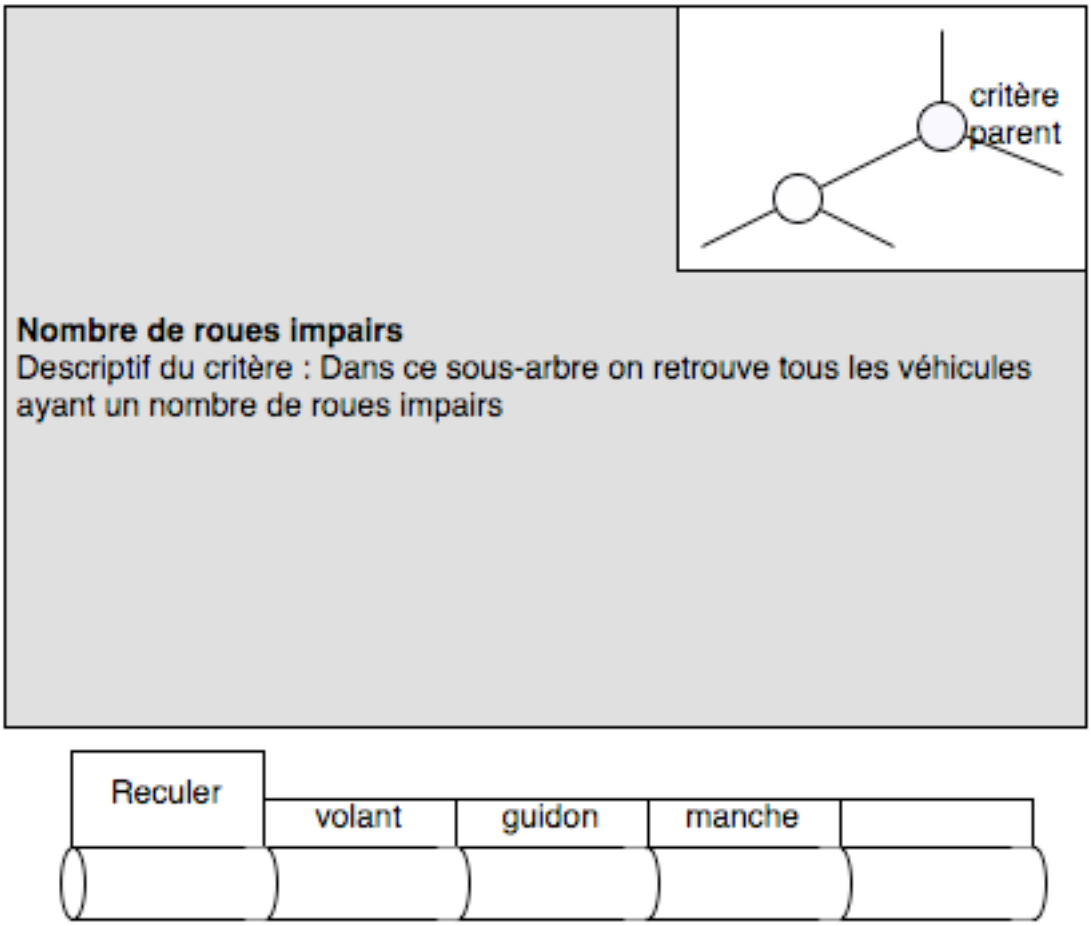


Figure 302 : Illustration de la composition spatiale proposée : Dans le coin supérieur droit est affichée l'arbre (objet de la tâche, Figure 297) et le critère parent (outil pour reculer, Figure 301). Au milieu est afficher le critère courant (objet de la tâche). En bas ou a l'horizontal se trouve l'outil pour reculer (Figure 301) juxtaposé avec l'outil pour avancer (Figure 299).

La solution présentée **Figure 303** concerne la tâche « se déplacer latéralement sur le critère frère de gauche/droite ». Il s'agit des onglets dédiés pour ces tâches.

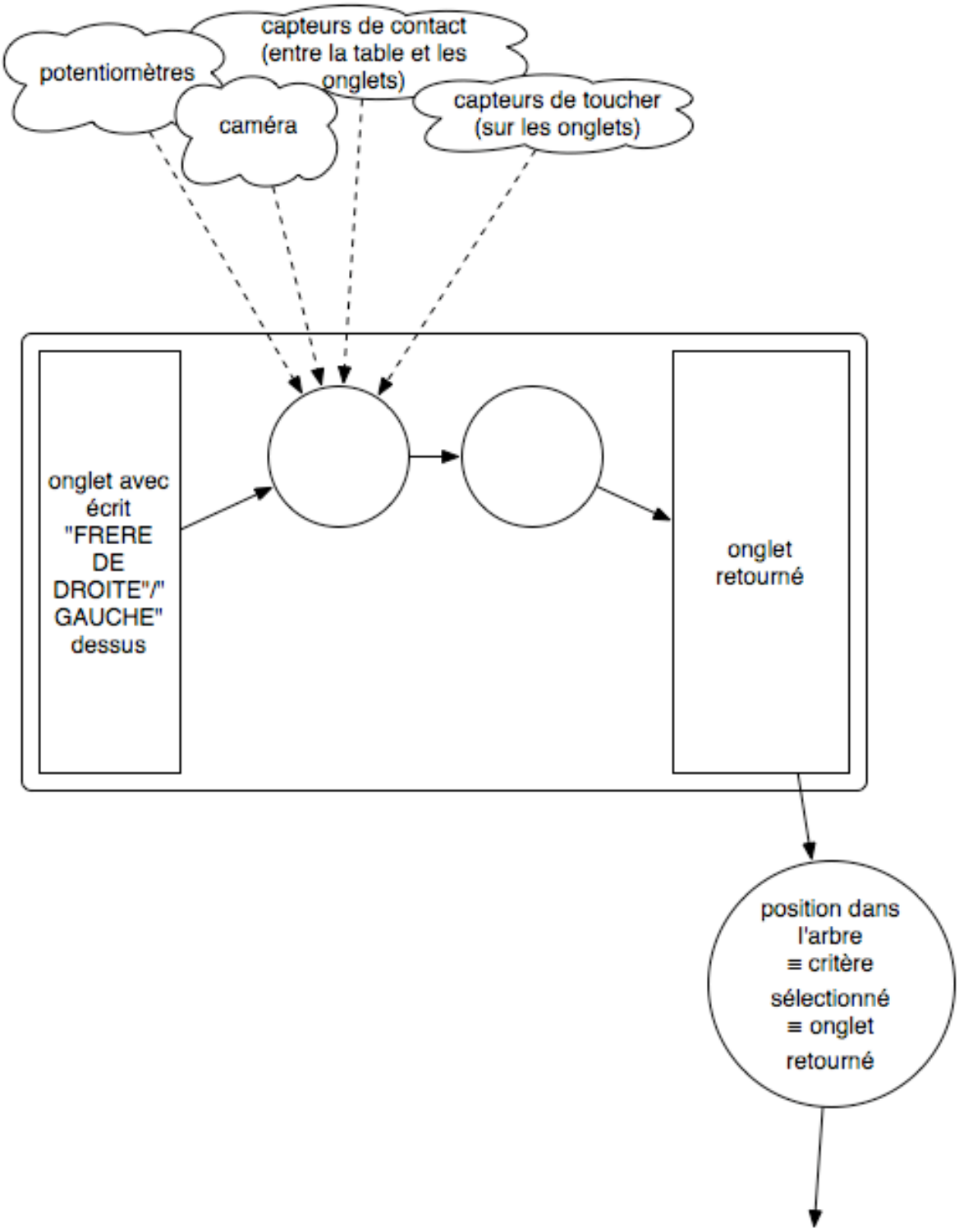


Figure 303 : Outil onglet pour se déplacer latéralement dans l'arbre (frère de gauche/droite).

Pour cette même tâche, l'outil présenté **Figure 304** a été pensé par analogie avec la machine à écrire quand on changeait de ligne : on peut déplacer la barre d'onglet sur la gauche ou sur la droite le long de son axe pour déplacer sa position dans l'arbre, et elle revient toute seule. Cette solution a seulement été partiellement conçue.

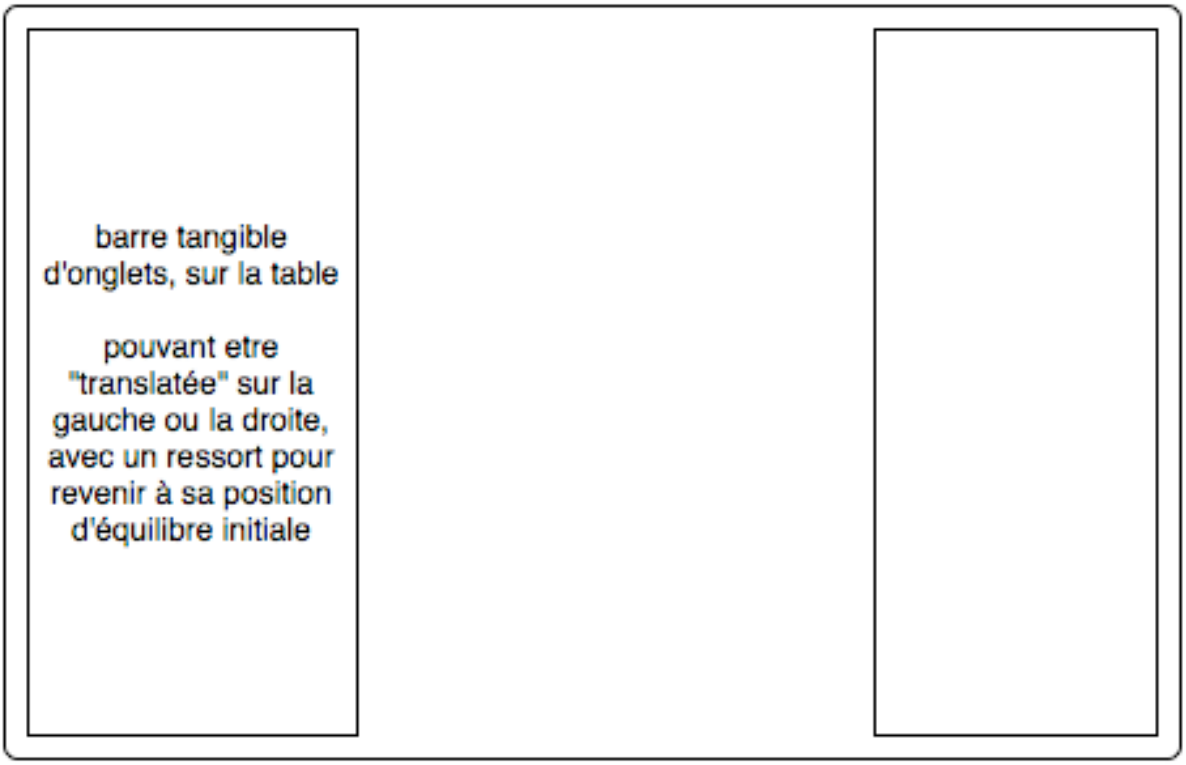


Figure 304 : Début de solution pour un outil pour se déplacer latéralement dans l'arbre (frère de gauche/droite).)

2.2. Levier

L'outil présenté **Figure 305** est analogue au levier de vitesse : l'utilisateur doit déplacer le levier pour explorer l'arbre. Deux possibilités ont été envisagées :

- Un levier dont la position absolue dans l'espace physique est captée,
- Un levier dont la position relative a la précédente (déplacement) est captée, de type souris/phantom/etc.

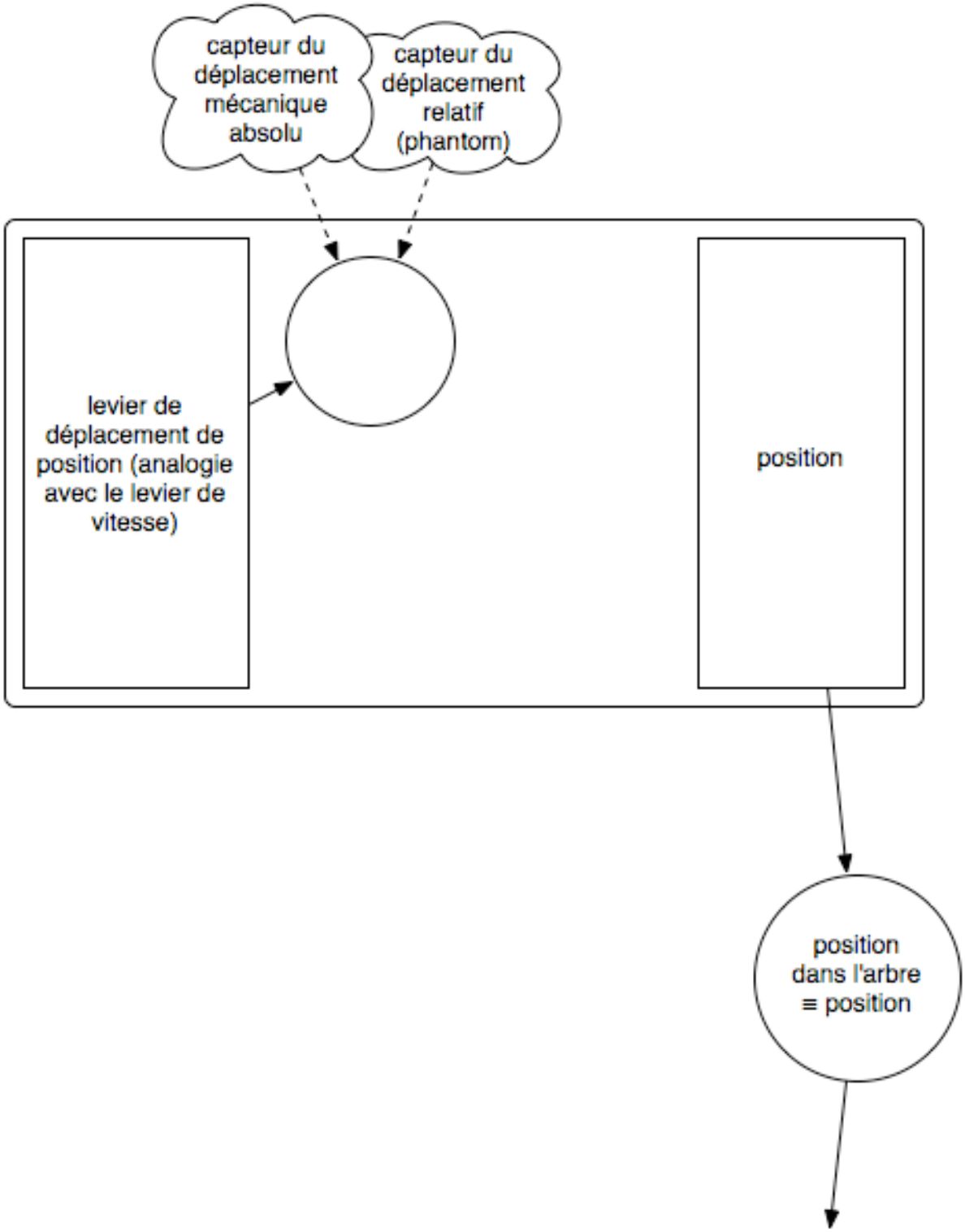


Figure 305 : Début de conception d'outil pour avancer dans un critère, reculer, et voir le frère de gauche ou droite du critère courant.

2.3. Outils à poser

L'outil présenté **Figure 306** peut prendre plusieurs formes physiques (avec ou sans métaphore de nom), mais dans tous les cas, il s'agit pour l'utilisateur de poser des objets tous le long du chemin parcouru dans l'arbre, ou de les retirer pour reculer. La partie physique de cet outil peut être des morceaux de mie de pain (analogie avec le Petit Poucet), un fil à dérouler le long du chemin (analogie avec le fil d'Ariane), des jetons dédiés pour chaque critère, à bien choisir avant de les poser (pas de métaphore), ou encore des échelles (analogie avec l'exploration d'un arbre).

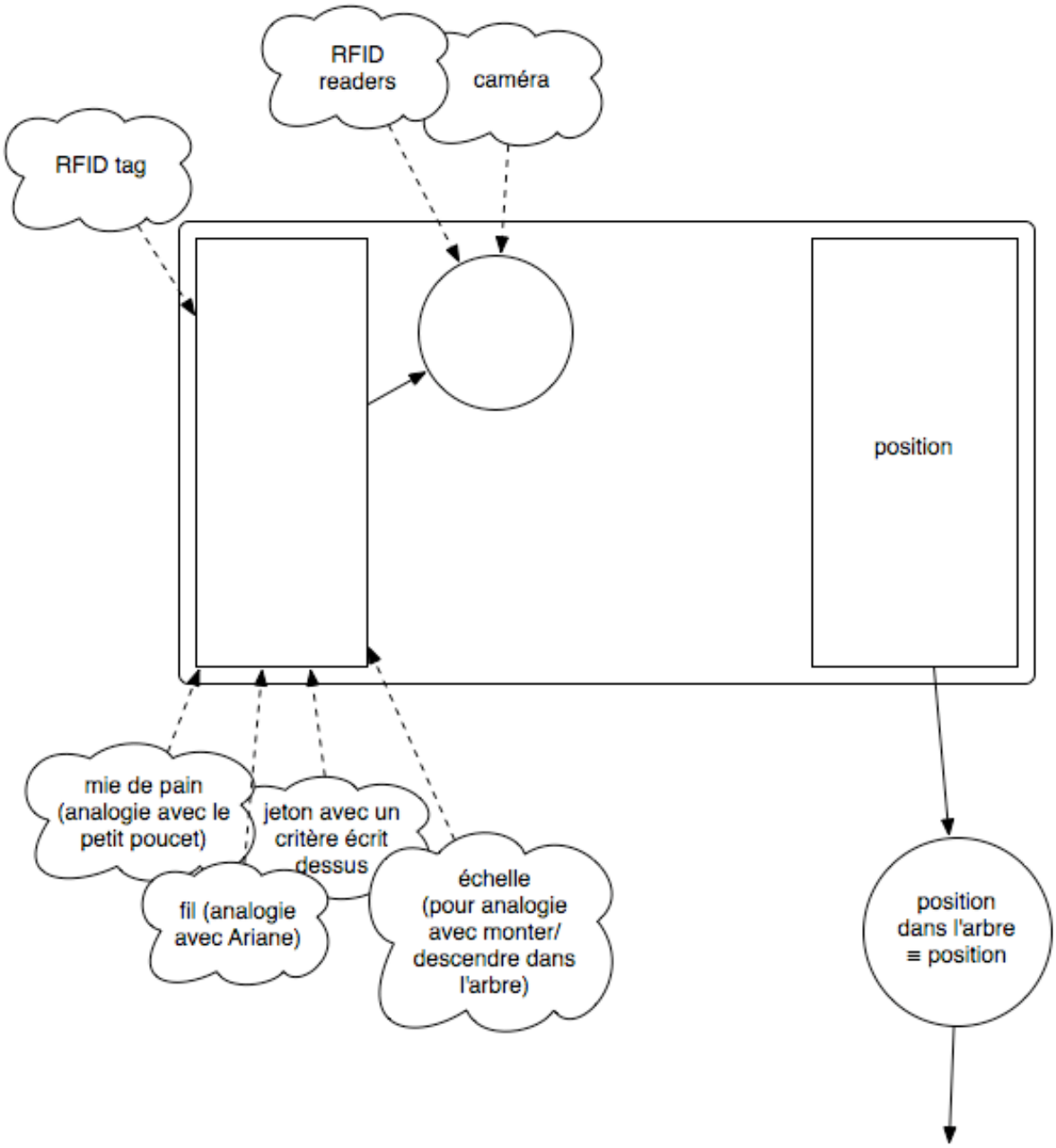


Figure 306 : Début de solution pour un outil pour avancer ou reculer.

Le dernier outil a été conçu pour interagir avec l'objet de la tâche modélisée à la **Figure 298**. Une partie élémentaire des cornes de l'animal (branche) est posée par l'utilisateur sur l'arbre pour le construire. Cette solution a seulement été partiellement conçue.

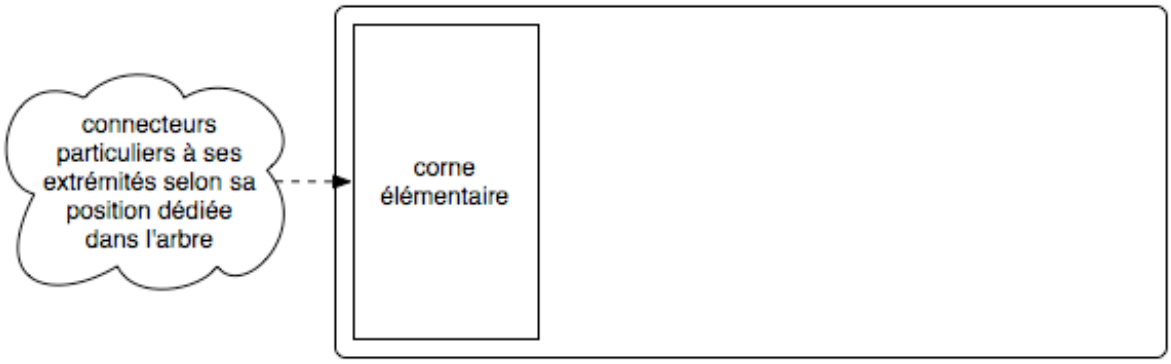


Figure 307 : Outil pour construire l'arbre, objet de la tâche de la Figure 298.

Annexe C : Questionnaires d'évaluation de la compréhensibilité

1. Questionnaire A





Questionnaire d'évaluation de notations dédiées à la conception d'interactions de nouvelle génération

Tout d'abord, nous tenons à vous remercier d'avoir accepté de répondre à ce questionnaire. Le sujet qui nous intéresse s'inscrit dans le cadre d'un projet de recherche financé par l'Agence Nationale de la Recherche (ANR). Le projet étudie les interfaces homme-machine de nouvelle génération, qui mélangent le monde de l'ordinateur avec le monde physique. Ce questionnaire a pour but de recueillir votre avis sur une nouvelle forme de schéma pour la conception de ces nouvelles formes d'interaction. Votre avis nous intéresse et contribuera à évaluer cette nouvelle notation et à l'améliorer.

Ce questionnaire a pour but d'évaluer les schémas proposés et non pas vos connaissances. Nous vous demandons de répondre le plus sincèrement possible, dans l'ordre de présentation des questions. Afin de ne pas biaiser les résultats, veuillez répondre le plus rapidement possible tout en évitant de vous tromper.

Nous estimons qu'environ 1h15 peut être nécessaire pour remplir ce questionnaire. Nous vous demandons de ne pas vous arrêter, et de le remplir d'une seule traite. À titre indicatif, merci de noter ci-après l'heure à laquelle vous commencez :

……h …… min

Pour commencer

1. Habituellement, lorsque vous avez besoin de concevoir un design, comment faites-vous ?

……...……
...……
……..
……...……
...……
……..
……...……
...……
……..
……...……
...……
……..

2. Lisez l'histoire ci-dessous :

1ère histoire

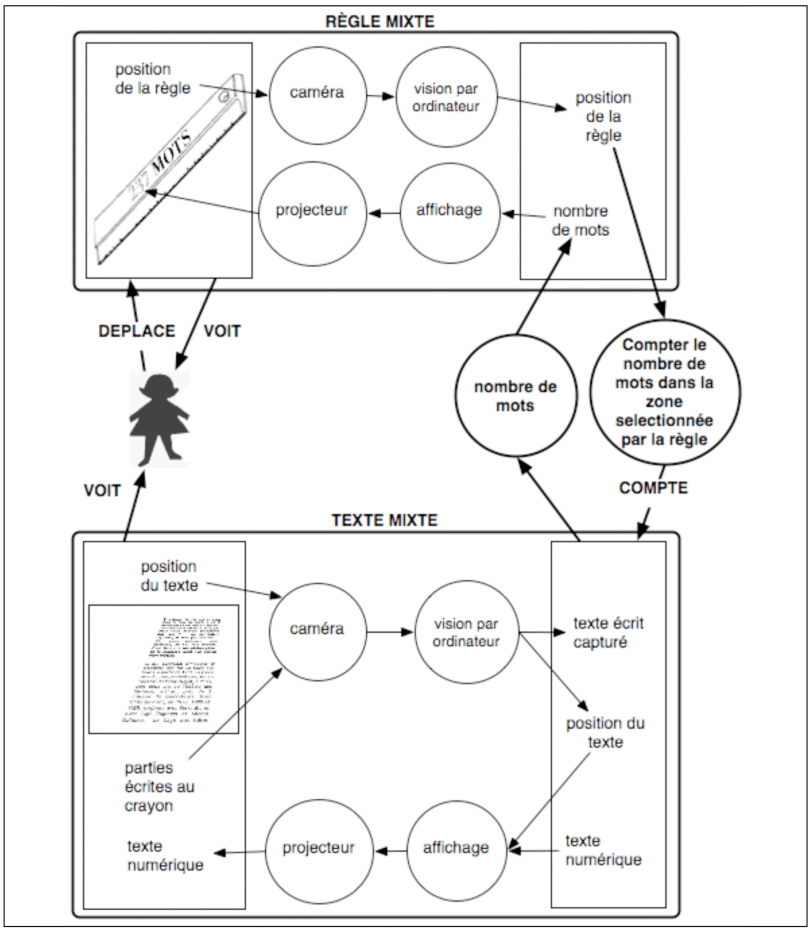
« Marie lit une page Web sur l'écran de son ordinateur. Le texte est en anglais, mais elle préférerait le lire en français. Elle se saisit alors de son dictionnaire Anglais-Français, mais au lieu de l'ouvrir, elle l'approche de son écran. Lorsque le dictionnaire est assez proche de l'écran, son navigateur Web ouvre alors une nouvelle fenêtre contenant la page traduite en français. Afin que le système identifie le dictionnaire, celui-ci contient un marqueur RFID dans la couverture, et l'écran de l'ordinateur de Marie contient un lecteur RFID. »

3. Sur la page suivante, dessinez librement un schéma pour décrire ce qui se passe dans le texte ci-dessus.

Première partie

4. Regardez ce premier schéma et essayez de comprendre ce que pourrait faire l'usager.

1er schéma



5. Pouvez-vous expliquer ce que décrit ce schéma? De la même manière que dans la 1ère histoire, vous ferez une description en quelques lignes de ce que vous avez compris :

……..……
..……
……..
……..……
..……
……..
……..……
..……
……..
……..……

……
……..
……...……
……
……..
……...……
……
……..

6. Sur une échelle allant de 1 à 10 quel est le niveau de confiance que **vous accordez à votre réponse** ?

1 **10**



Tout à fait sûr(e) Pas du tout sûr(e)

Pourquoi ?

……...……
……
……..

7. Pour vous, répondre a été …

Très facile	Plutôt facile	Plutôt difficile	Très difficile
☐	☐	☐	☐

Pourquoi ?

……...……
……
……..

8. Avez-vous répondu …

Très rapidement	Plutôt rapidement	Plutôt lentement	Très lentement
☐	☐	☐	☐

9. Sur une échelle de compréhension allant de 1 à 10, quelle note attribueriez-vous à ce type de schéma ?

1 **10**

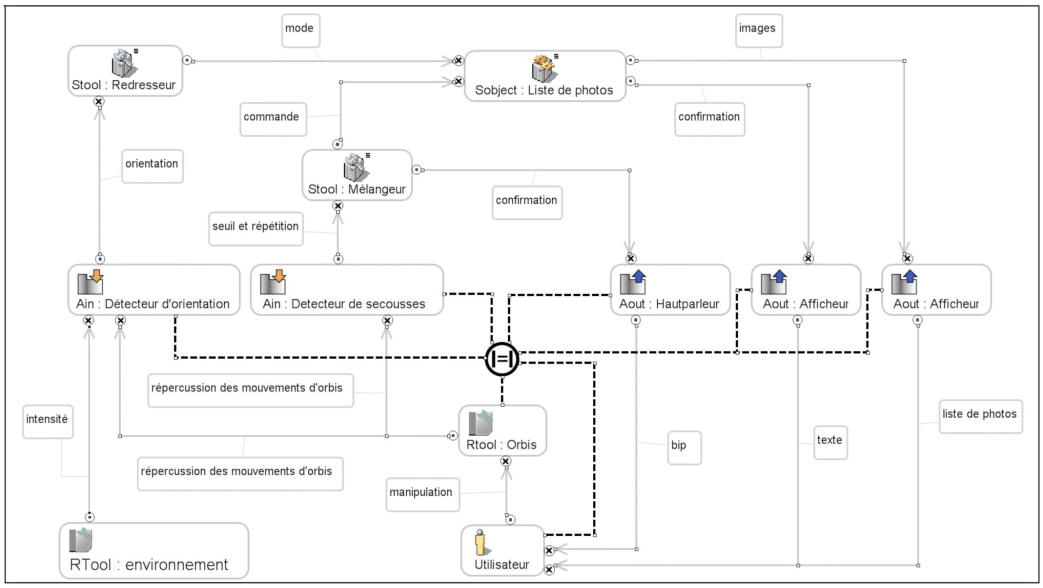


Pas du tout compréhensible Tout à fait compréhensible

Deuxième partie

10. Regardez ce deuxième schéma et essayez de comprendre ce que pourrait faire l'usager.

2ème schéma



11. Pouvez-vous expliquer ce que décrit ce schéma? De la même manière que dans la 1ère histoire, vous ferez une description en quelques lignes de ce que vous avez compris :

……...……
……
……
……...……
……
……
……...……
……
……
……...……
……
……
……...……
……
……
……...……
……
……

12. Sur une échelle allant de 1 à 10 quel est le niveau de confiance que **vous accordez à votre réponse** ?

1 **10**



Tout à fait sûr(e) Pas du tout sûr(e)

Pourquoi ?
……...……
..……
……..

13. Pour vous, répondre a été …

Très facile	Plutôt facile	Plutôt difficile	Très difficile
☐	☐	☐	☐

Pourquoi ?
……...……
..……
……..

14. Avez-vous répondu …

Très rapidement	Plutôt rapidement	Plutôt lentement	Très lentement
☐	☐	☐	☐

15. Sur une échelle de compréhension allant de 1 à 10, quelle note attribueriez-vous à ce type de schéma ?

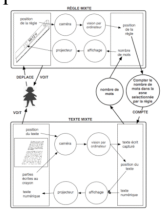
1 **10**



Pas du tout compréhensible Tout à fait compréhensible

Troisième Partie

16. Si vous deviez décrire un système interactif, parmi les deux types de schémas que vous venez de voir, lequel choisiriez-vous ? (1 seule réponse)

☐ Le premier : 

☐ Le deuxième : 

☐ Aucun

Pourquoi ?
……...……
..……
……..

17. Si vous avez choisi le premier ou le deuxième, est-ce que vous le conseilleriez à d'autres personnes pour décrire un système interactif ?

Oui, tout à fait	Oui, plutôt	Non, plutôt pas	Non, pas du tout
☐	☐	☐	☐

Pourquoi ?
……...……
..……
……..

18. Est-ce que voir les éléments graphiques suivants du premier schéma vous aide ?

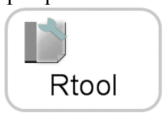




Oui, tout à fait	Oui, plutôt	Non, plutôt pas	Non, pas du tout
☐	☐	☐	☐

Pourquoi ?
……...……
..……
……..

19. Est-ce que voir les éléments graphiques suivants du deuxième schéma vous aide ?



Oui, tout à fait	Oui, plutôt	Non, plutôt pas	Non, pas du tout
☐	☐	☐	☐

Pourquoi ?
……...……
..……
……..

20. Dans certaines parties de schémas, il y a :
    a. Soit des éléments écrits : « Orbis »
    b. Soit des éléments dessinés : 

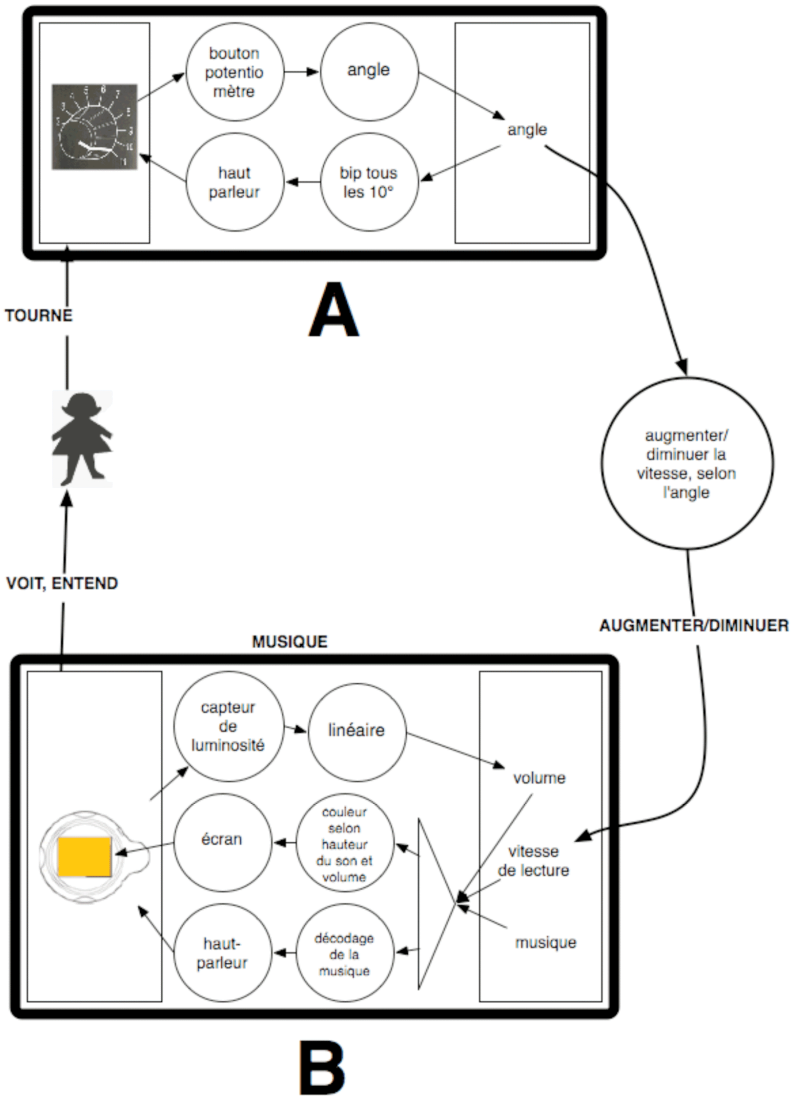
Que préférez-vous?

- ☐ Je préfère les éléments écrits
- ☐ Je préfère les éléments dessinés
- ☐ Je préfèrerais avoir les deux
- ☐ Cela n'a pas d'importance

Pourquoi ?
……...……
..……
……..

Quatrième Partie

Maintenant, nous allons nous intéresser à la représentation graphique suivante :



21. Pour vous, que représentent les rectangles aux coins arrondis ?

☐ Des objets à moitié physique et à moitié numérique

☐ Des éléments physiques

☐ Des éléments numériques

☐ Des appareillages

☐ Des zones dans la salle où se déroule l'interaction

22. Pour vous, quel est le rôle des blocs A et B ?

☐ A est un objet pour l'interaction de l'utilisateur vers l'ordinateur et B est un objet pour l'interaction de l'ordinateur vers l'utilisateur

☐ A est un objet pour l'interaction de l'ordinateur vers l'utilisateur et B est un objet pour l'interaction de l'utilisateur vers l'ordinateur

☐ A est un outil et B est l'objet cible de l'utilisateur

☐ A est l'objet cible de l'utilisateur et B est un outil

☐ A et B représentent des tables physiques sur lesquelles sont posés des éléments manipulés pour réaliser l'interaction

23. Pour vous, quels sont les avantages et les inconvénients de la séparation des blocs A et B ?

……...……
……..
……...……
...……
……...……
……...……
……..
……...……
...……
……...……

24. Pour vous, que représentent les rectangles aux coins droits ?

☐ Des représentations concrètes ou des représentations abstraites

☐ Des données physiques ou numériques

☐ Des parties physiques ou numériques

☐ Des objets tangibles

☐ Des dispositifs capteurs ou émetteurs

25. Pour vous, que représentent les cercles?

☐ Un moyen de communication

☐ Une transformation

☐ Une fonctionnalité

☐ Un dispositif

☐ Une donnée

26. Pour vous, quelle est la différence entre les cercles qui sont à l'intérieur des rectangles aux coins arrondis et celui qui est à l'extérieur ?

☐ Ceux qui sont à l'intérieur sont des dispositifs, ceux qui sont à l'extérieur sont des ordinateurs

☐ Ceux qui sont à l'intérieur sont des fonctionnalités de l'objet, ceux qui sont à l'extérieur sont une fonctionnalité globale du système

☐ Ceux qui sont à l'intérieur font le lien entre les parties physiques et numériques de l'objet, ceux qui sont à l'extérieur représentent l'interaction entre les objets

☐ Il n'y a pas de différence

☐ Ceux qui sont à l'intérieur ne sont pas placé au même endroit dans l'espace physique où se déroule l'interaction, que ceux qui sont à l'extérieur

27. Pour vous, que représente le triangle ?

☐ Une passerelle

☐ La décomposition/séparation/fission de l'information

☐ Un filtre de l'information

☐ Un transfert sur réseau informatique

☐ Une proximité physique des éléments à l'origine de la flèche

28. Pour vous, que représentent les flèches ?

☐ Des flux d'informations

☐ Des connectiques physiques (câbles)

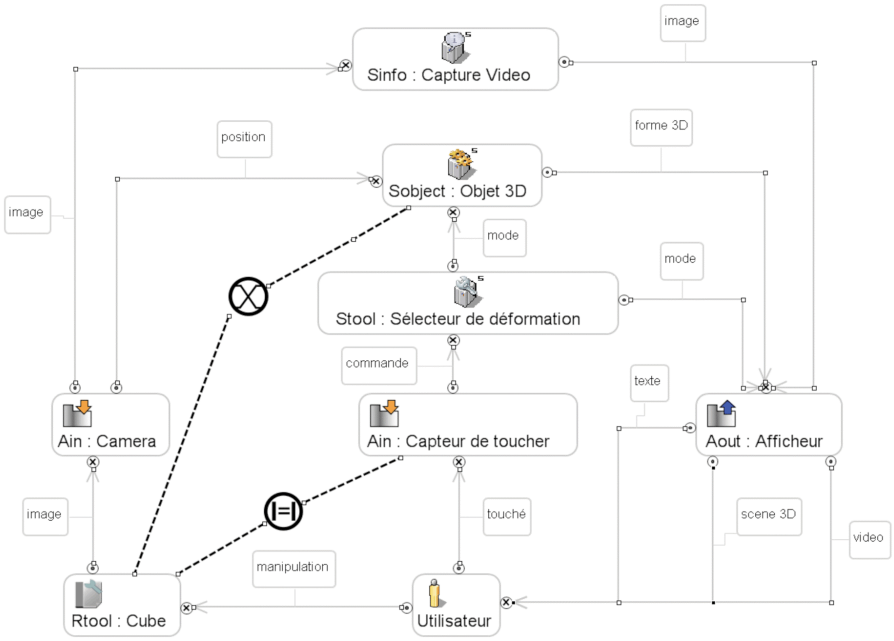
☐ Des retours d'informations

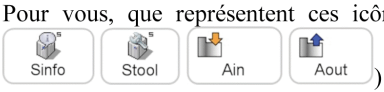
☐ L'ordre temporel du déroulement de l'action

☐ Un sens de lecture du diagramme

Cinquième Partie

Maintenant, nous allons nous intéresser à la représentation graphique suivante :



29. Pour vous, que représentent ces icônes en général ? ( )

□ Des étapes du déroulement de l'interaction

□ Des emplacements dans la scène interactive

□ Des fonctionnalités du système

□ Des entités du système

□ Des outils

30. Pour vous, quelle est la différence entre ces 2 ensembles d'icônes ?

1) 

2) 

□ Les (1) sont des éléments d'interaction en entrée et les (2) sont des éléments d'interaction en sortie

□ Les (1) sont des éléments physiques et les (2) sont des éléments numériques

□ Aucune différence, ils constituent un même ensemble

□ Les (1) sont des éléments concrets et les (2) sont des éléments abstraits

□ Les (1) sont des éléments mixtes (mêlant physiques et numériques) et les (2) sont des éléments non mixtes (donc uniquement physiques ou numériques)

31. Pour vous, quelle est la différence entre l'élément représenté par l'icône 

et l'élément représenté par l'icône 

?

……..……
..……
..……
..……
..……
..……
..……
..……
……..
……..……

32. Pour vous, que permet d'exprimer cet élément  ?

□ La ressemblance entre éléments

□ La proximité physique entre éléments

□ La comptabilité technologique

□ L'analogie entre un élément physique et un élément numérique

□ Une communication sans-fil entre éléments

33. Pour vous, que permet d'exprimer cet élément  ?

□ La proximité physique entre éléments

□ La comptabilité technologique

□ Un objet composé de plusieurs éléments

□ L'analogie entre un élément physique et un élément numérique

□ Un raccourci d'interaction

34. Pour vous, que représentent une flèche comme celle là  ?

□ Des connectiques physiques (câbles)
□ Une connexion d'éléments
□ Un flux d'informations
□ Une transition entre 2 états du système
□ Des feedbacks

35. Pour vous, que représentent les annotations attachées aux flèches ?

□ Des données
□ Une propriété utilisée pour transmettre l'information
□ Le langage utilisé pour coder l'information
□ Des fonctionnalités du système
□ Des indications quelconques

Renseignements

36. Quelle est votre année de naissance ? ……..……

37. Vous êtes :
    - ☐ Une femme
    - ☐ Un homme

38. Par quelle expertise pourriez-vous être ou êtes-vous effectivement conduit à concevoir des interfaces ?
    - ☐ Ergonomie
    - ☐ Design
    - ☐ Informatique
    - ☐ Arts des Nouveaux Médias
    - ☐ Autre : ……..

39. Depuis combien de temps exercez-vous votre métier ? …….

40. Quelle connaissance avez-vous du domaine de l'interaction homme-machine en général ?

Aucune	Peu expérimenté	Plutôt expérimenté	Expert
☐	☐	☐	☐

41. Quelle connaissance avez-vous du domaine de systèmes mixtes en particulier ?

Aucune	Peu expérimenté	Plutôt expérimenté	Expert
☐	☐	☐	☐

Pour connaître le temps dont vous avez eu besoin, merci de noter ci-après l'heure à laquelle vous terminez :

……h …… min

Merci !

2. Questionnaire B





Questionnaire d'évaluation de notations dédiées à la conception d'interactions de nouvelle génération

Tout d'abord, nous tenons à vous remercier d'avoir accepté de répondre à ce questionnaire. Le sujet qui nous intéresse s'inscrit dans le cadre d'un projet de recherche financé par l'Agence Nationale de la Recherche (ANR). Le projet étudie les interfaces homme-machine de nouvelle génération, qui mélangent le monde de l'ordinateur avec le monde physique. Ce questionnaire a pour but de recueillir votre avis sur une nouvelle forme de schéma pour la conception de ces nouvelles formes d'interaction. Votre avis nous intéresse et contribuera à évaluer cette nouvelle notation et à l'améliorer.

Ce questionnaire a pour but d'évaluer les schémas proposés et non pas vos connaissances. Nous vous demandons de répondre le plus sincèrement possible, dans l'ordre de présentation des questions. Afin de ne pas biaiser les résultats, veuillez répondre le plus rapidement possible tout en évitant de vous tromper.

Nous estimons qu'environ 1h15 peut être nécessaire pour remplir ce questionnaire. Nous vous demandons de ne pas vous arrêter, et de le remplir d'une seule traite. À titre indicatif, merci de noter ci-après l'heure à laquelle vous commencez :

……h …… min

Pour commencer

1. Habituellement, lorsque vous avez besoin de concevoir un design, comment faites-vous ?

……...……
...……
……..
……...……
...……
……..
……...……
...……
……..
……...……
...……
……..

2. Lisez l'histoire ci-dessous :

1ère histoire

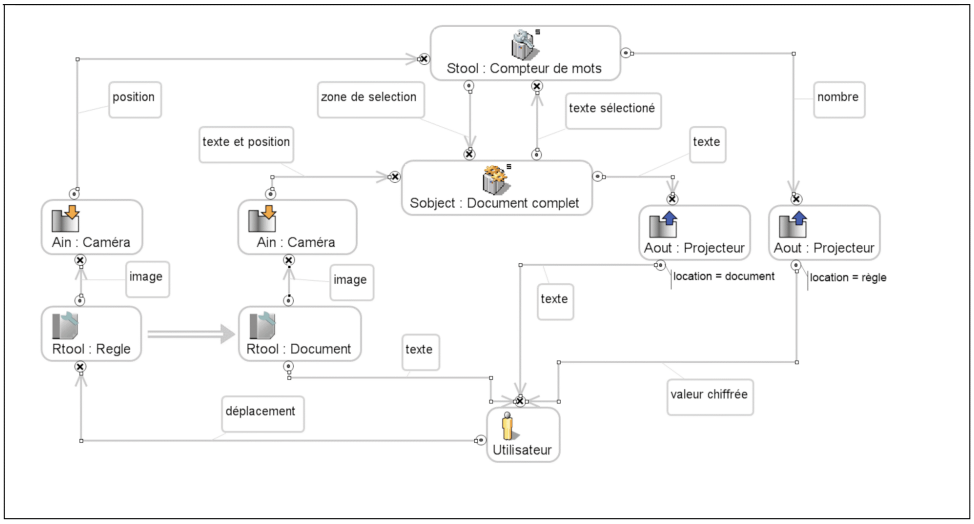
« Marie lit une page Web sur l'écran de son ordinateur. Le texte est en anglais, mais elle préférerait le lire en français. Elle se saisit alors de son dictionnaire Anglais-Français, mais au lieu de l'ouvrir, elle l'approche de son écran. Lorsque le dictionnaire est assez proche de l'écran, son navigateur Web ouvre alors une nouvelle fenêtre contenant la page traduite en français. Afin que le système identifie le dictionnaire, celui-ci contient un marqueur RFID dans la couverture, et l'écran de l'ordinateur de Marie contient un lecteur RFID. »

3. Sur la page suivante, dessinez librement un schéma pour décrire ce qui se passe dans le texte ci-dessus.

Première partie

4. Regardez ce premier schéma et essayez de comprendre ce que pourrait faire l'usager.

1er schéma



5. Pouvez-vous expliquer ce que décrit ce schéma? De la même manière que dans la 1ère histoire, vous ferez une description en quelques lignes de ce que vous avez compris :

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

6. Sur une échelle allant de 1 à 10 quel est le niveau de confiance que **vous accordez à votre réponse** ?

1 **10**



Tout à fait sûr(e) Pas du tout sûr(e)

Pourquoi ?
……...……
..……
……..

7. Pour vous, répondre a été …

Très facile	Plutôt facile	Plutôt difficile	Très difficile
☐	☐	☐	☐

Pourquoi ?
……...……
..……
……..

8. Avez-vous répondu …

Très rapidement	Plutôt rapidement	Plutôt lentement	Très lentement
☐	☐	☐	☐

9. Sur une échelle de compréhension allant de 1 à 10, quelle note attribueriez-vous à ce type de schéma ?

1 **10**

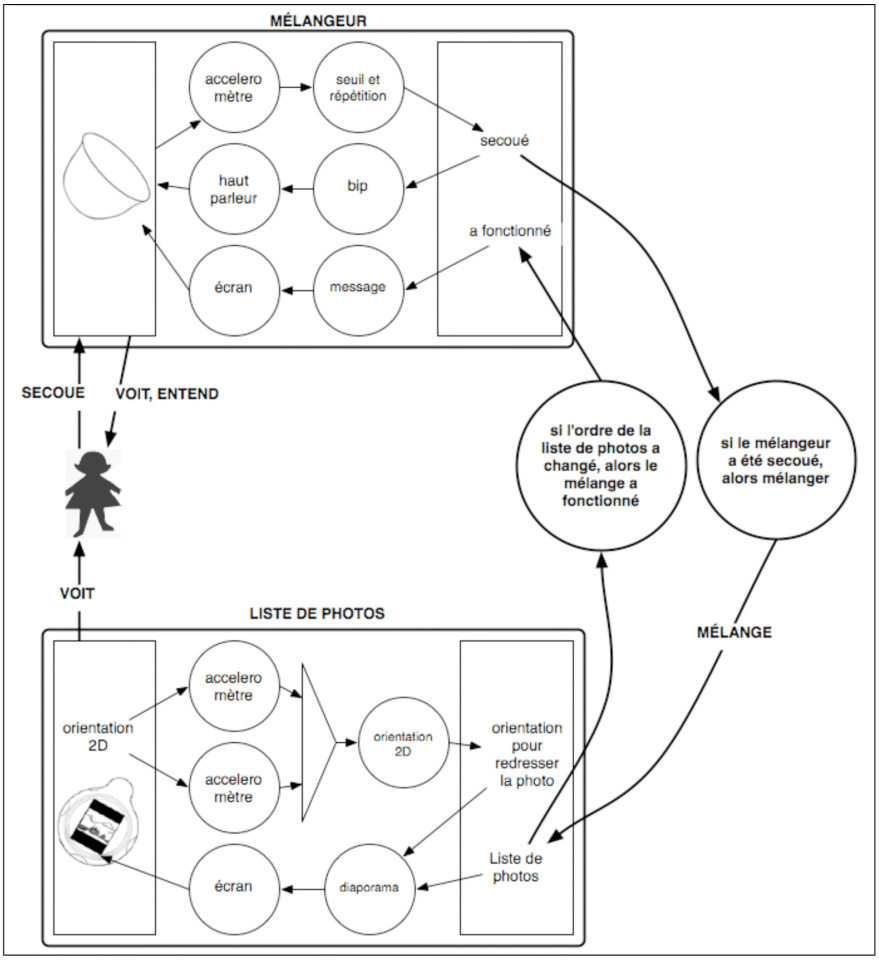


Pas du tout compréhensible Tout à fait compréhensible

Deuxième partie

10. Regardez ce deuxième schéma et essayez de comprendre ce que pourrait faire l'usager.

2ème schéma



11. Pouvez-vous expliquer ce que décrit ce schéma? De la même manière que dans la 1ère histoire, vous ferez une description en quelques lignes de ce que vous avez compris :

……...……
...……
……..
……...……
...……
……..
……...……
...……

……..
……..……
..……
……..
……..……
..……
……..
……..……
..……
……..

12. Sur une échelle allant de 1 à 10 quel est le niveau de confiance que **vous accordez à votre réponse** ?

1 **10**



Tout à fait sûr(e) Pas du tout sûr(e)

Pourquoi ?

……..……
..……
……..

13. Pour vous, répondre a été …

Très facile	Plutôt facile	Plutôt difficile	Très difficile
☐	☐	☐	☐

Pourquoi ?

……..……
..……
……..

14. Avez-vous répondu …

Très rapidement	Plutôt rapidement	Plutôt lentement	Très lentement
☐	☐	☐	☐

15. Sur une échelle de compréhension allant de 1 à 10, quelle note attribueriez-vous à ce type de schéma ?

1 **10**

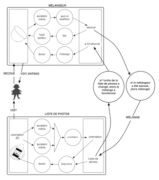


Pas du tout compréhensible Tout à fait compréhensible

Troisième Partie

16. Si vous deviez décrire un système interactif, parmi les deux types de schémas que vous venez de voir, lequel choisiriez-vous ? (1 seule réponse)

☐ Le premier : 

☐ Le deuxième : 

☐ Aucun

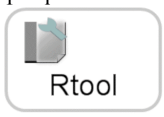
Pourquoi ?
……...……
...……
……..

17. Si vous avez choisi le premier ou le deuxième, est-ce que vous le conseilleriez à d'autres personnes pour décrire un système interactif ?

Oui, tout à fait	Oui, plutôt	Non, plutôt pas	Non, pas du tout
☐	☐	☐	☐

Pourquoi ?
……...……
...……
……..

18. Est-ce que voir les éléments graphiques suivants du premier schéma vous aide ?



Oui, tout à fait	Oui, plutôt	Non, plutôt pas	Non, pas du tout
☐	☐	☐	☐

Pourquoi ?
……...……
...……
……..

19. Est-ce que voir les éléments graphiques suivants du deuxième schéma vous aide ?



Oui, tout à fait ☐ Oui, plutôt ☐ Non, plutôt pas ☐ Non, pas du tout ☐

Pourquoi ?
……...……
..……
……..

20. Dans certaines parties de schémas, il y a :
    a. Soit des éléments écrits : « Règle », « Document »
    b. Soit des éléments dessinés : 

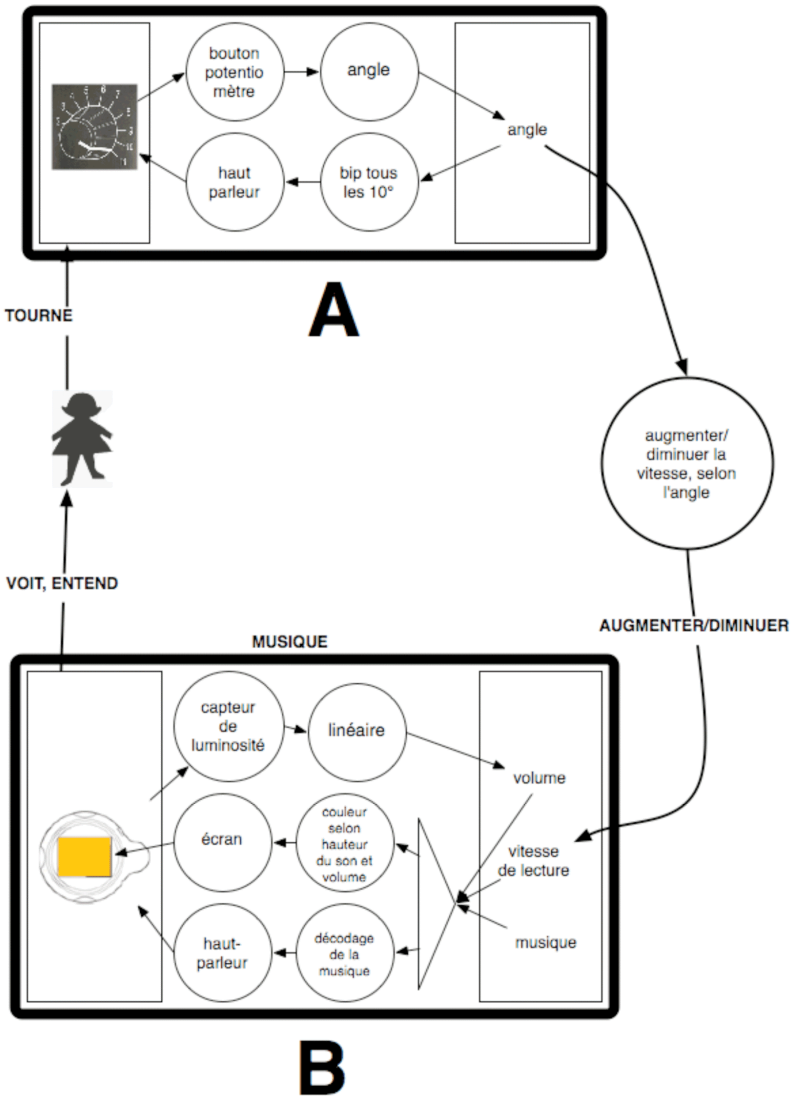
Que préférez-vous?

- ☐ Je préfère les éléments écrits
- ☐ Je préfère les éléments dessinés
- ☐ Je préfèrerais avoir les deux
- ☐ Cela n'a pas d'importance

Pourquoi ?
……...……
..……
……..

Quatrième Partie

Maintenant, nous allons nous intéresser à la représentation graphique suivante :



21. Pour vous, que représentent les rectangles aux coins arrondis ?

☐ Des objets à moitié physique et à moitié numérique

☐ Des éléments physiques

☐ Des éléments numériques

☐ Des appareillages

☐ Des zones dans la salle où se déroule l'interaction

22. Pour vous, quel est le rôle des blocs A et B ?

☐ A est un objet pour l'interaction de l'utilisateur vers l'ordinateur et B est un objet pour l'interaction de l'ordinateur vers l'utilisateur

☐ A est un objet pour l'interaction de l'ordinateur vers l'utilisateur et B est un objet pour l'interaction de l'utilisateur vers l'ordinateur

☐ A est un outil et B est l'objet cible de l'utilisateur

☐ A est l'objet cible de l'utilisateur et B est un outil

☐ A et B représentent des tables physiques sur lesquelles sont posés des éléments manipulés pour réaliser l'interaction

23. Pour vous, quels sont les avantages et les inconvénients de la séparation des blocs A et B ?

……...……
……..
……...……
...……
……...……
……...……
……..
……...……
...……
……...……

24. Pour vous, que représentent les rectangles aux coins droits ?

☐ Des représentations concrètes ou des représentations abstraites

☐ Des données physiques ou numériques

☐ Des parties physiques ou numériques

☐ Des objets tangibles

☐ Des dispositifs capteurs ou émetteurs

25. Pour vous, que représentent les cercles?

☐ Un moyen de communication

☐ Une transformation

☐ Une fonctionnalité

☐ Un dispositif

☐ Une donnée

26. Pour vous, quelle est la différence entre les cercles qui sont à l'intérieur des rectangles aux coins arrondis et celui qui est à l'extérieur ?

☐ Ceux qui sont à l'intérieur sont des dispositifs, ceux qui sont à l'extérieur sont des ordinateurs

☐ Ceux qui sont à l'intérieur sont des fonctionnalités de l'objet, ceux qui sont à l'extérieur sont une fonctionnalité globale du système

☐ Ceux qui sont à l'intérieur font le lien entre les parties physiques et numériques de l'objet, ceux qui sont à l'extérieur représentent l'interaction entre les objets

☐ Il n'y a pas de différence

☐ Ceux qui sont à l'intérieur ne sont pas placé au même endroit dans l'espace physique où se déroule l'interaction, que ceux qui sont à l'extérieur

27. Pour vous, que représente le triangle ?

☐ Une passerelle

☐ La décomposition/séparation/fission de l'information

☐ Un filtre de l'information

☐ Un transfert sur réseau informatique

☐ Une proximité physique des éléments à l'origine de la flèche

28. Pour vous, que représentent les flèches ?

☐ Des flux d'informations

☐ Des connectiques physiques (câbles)

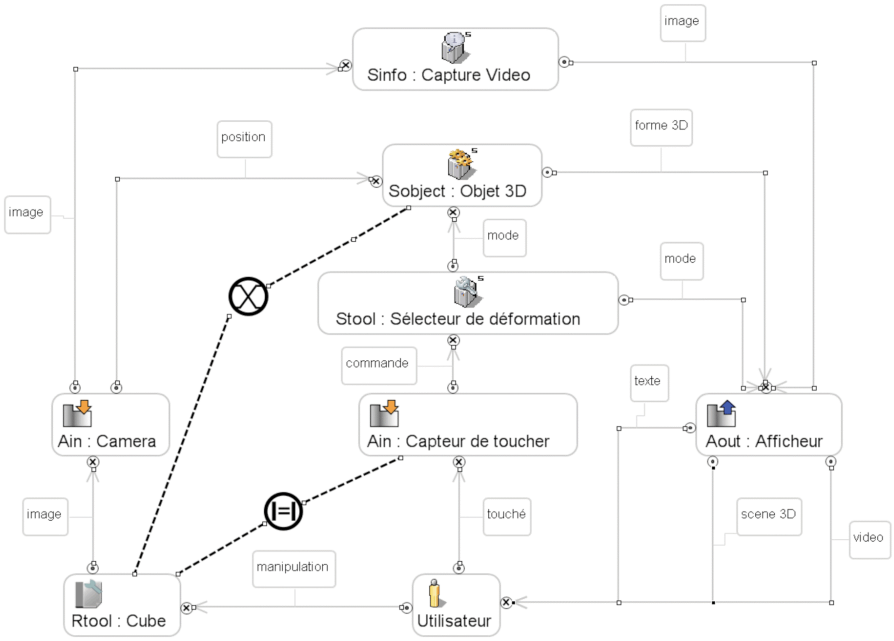
☐ Des retours d'informations

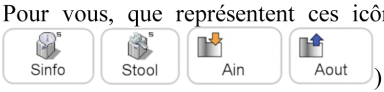
☐ L'ordre temporel du déroulement de l'action

☐ Un sens de lecture du diagramme

Cinquième Partie

Maintenant, nous allons nous intéresser à la représentation graphique suivante :



29. Pour vous, que représentent ces icônes en général ? ( )

□ Des étapes du déroulement de l'interaction

□ Des emplacements dans la scène interactive

□ Des fonctionnalités du système

□ Des entités du système

□ Des outils

30. Pour vous, quelle est la différence entre ces 2 ensembles d'icônes ?

1) 

2) 

□ Les (1) sont des éléments d'interaction en entrée et les (2) sont des éléments d'interaction en sortie

□ Les (1) sont des éléments physiques et les (2) sont des éléments numériques

□ Aucune différence, ils constituent un même ensemble

□ Les (1) sont des éléments concrets et les (2) sont des éléments abstraits

□ Les (1) sont des éléments mixtes (mêlant physiques et numériques) et les (2) sont des éléments non mixtes (donc uniquement physiques ou numériques)

31. Pour vous, quelle est la différence entre l'élément représenté par l'icône 

et l'élément représenté par l'icône 

?

……...……
...……
...……
...……
...……
...……
...……
...……
……..
……...……

32. Pour vous, que permet d'exprimer cet élément  ?

□ La ressemblance entre éléments

□ La proximité physique entre éléments

□ La comptabilité technologique

□ L'analogie entre un élément physique et un élément numérique

□ Une communication sans-fil entre éléments

33. Pour vous, que permet d'exprimer cet élément  ?

□ La proximité physique entre éléments

□ La comptabilité technologique

□ Un objet composé de plusieurs éléments

□ L'analogie entre un élément physique et un élément numérique

□ Un raccourci d'interaction

34. Pour vous, que représentent une flèche comme celle là  ?

□ Des connectiques physiques (câbles)
□ Une connexion d'éléments
□ Un flux d'informations
□ Une transition entre 2 états du système
□ Des feedbacks

35. Pour vous, que représentent les annotations attachées aux flèches ?

□ Des données
□ Une propriété utilisée pour transmettre l'information
□ Le langage utilisé pour coder l'information
□ Des fonctionnalités du système
□ Des indications quelconques

Renseignements

36. Quelle est votre année de naissance ? ……..……

37. Vous êtes :
    - ☐ Une femme
    - ☐ Un homme

38. Par quelle expertise pourriez-vous être ou êtes-vous effectivement conduit à concevoir des interfaces ?
    - ☐ Ergonomie
    - ☐ Design
    - ☐ Informatique
    - ☐ Arts des Nouveaux Médias
    - ☐ Autre : ……..

39. Depuis combien de temps exercez-vous votre métier ? …….

40. Quelle connaissance avez-vous du domaine de l'interaction homme-machine en général ?

Aucune	Peu expérimenté	Plutôt expérimenté	Expert
☐	☐	☐	☐

41. Quelle connaissance avez-vous du domaine de systèmes mixtes en particulier ?

Aucune	Peu expérimenté	Plutôt expérimenté	Expert
☐	☐	☐	☐

Pour connaître le temps dont vous avez eu besoin, merci de noter ci-après l'heure à laquelle vous terminez :

……h …… min

Merci !

Annexe D : Spécifications détaillées de OP

1. RampInputLanguage

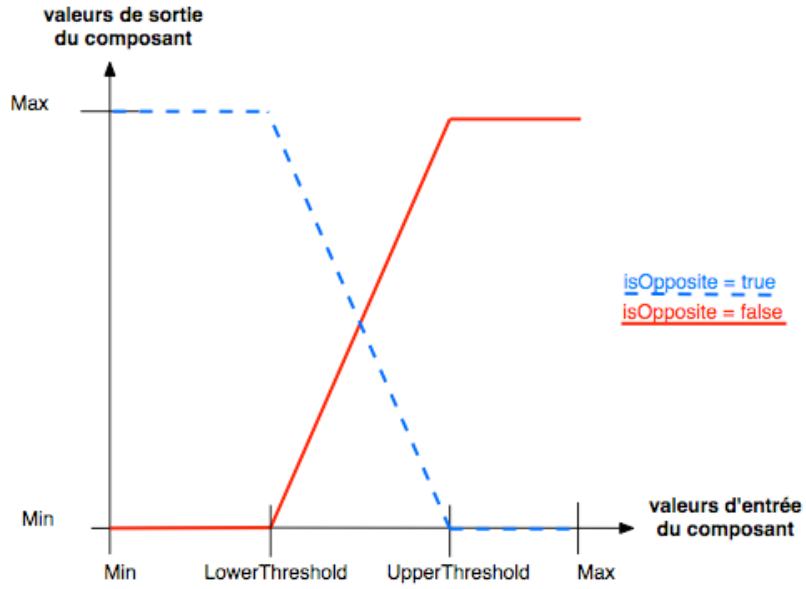


Figure 308 : Transformation effectuée par le composant *RampInputLanguage*.

La Figure 204 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 204, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pIsOpposite* (Figure 204 ligne 2) indique si le composant renvoie les valeurs dans leur sens originel (Figure 203 en pointillés bleus) ou dans l'autre sens (Figure 203 en ligne pleine verte).
3. Le paramètre *pMin* (Figure 204 ligne 3 et Figure 203) est un paramètre du constructeur qui permet d'indiquer au composant la valeur minimum reçue en entrée (et par conséquent envoyée en sortie).
4. Le paramètre *pLowerThreshold* (Figure 204 ligne 4 et Figure 203 en abscisse) permet de fixer le seuil en dessous duquel les valeurs seront « ignorées » (le composant renvoie *min*).
5. Le paramètre *pUpperThreshold* (Figure 204 ligne 5 et Figure 203 en abscisse) permet de fixer le seuil au-dessus duquel les valeurs seront « ignorées » (le composant renvoie *max*).
6. Le paramètre *pMax* (Figure 204 ligne 6 et Figure 203) est un paramètre du constructeur qui permet d'indiquer au composant la valeur maximum reçue en entrée (et par conséquent envoyée en sortie).

```
1 | RampInputLanguage(char* pName,
2 |                   bool pIsOpposite,
3 |                   int hardMin,
4 |                   int pLowerThreshold,
5 |                   int pUpperThreshold,
6 |                   int hardMax);
```

Figure 309 : Constructeur pour créer un RampInputLanguage.

Pour connecter ce composant, OP fournit des signaux et un *slot*. Pour connecter un dispositif d'entrée à ce composant, alors le *slot* proposé est :

```
void update(int pMessage, QTime pTime);
```

Le premier paramètre est la valeur entière reçue en entrée, le second est une estampille temporelle. Pour connecter ce composant à un autre composant, comme à une propriété numérique, les signaux proposés sont :

```
void updated(int pMessage, QTime pTime);
```

```
void updated(QVariant pMessage, QTime pTime);
```

Pour les deux signaux, le premier paramètre indique la valeur entière renvoyée par le composant, le second est une estampille temporelle.

2. ThresholdInputLanguage

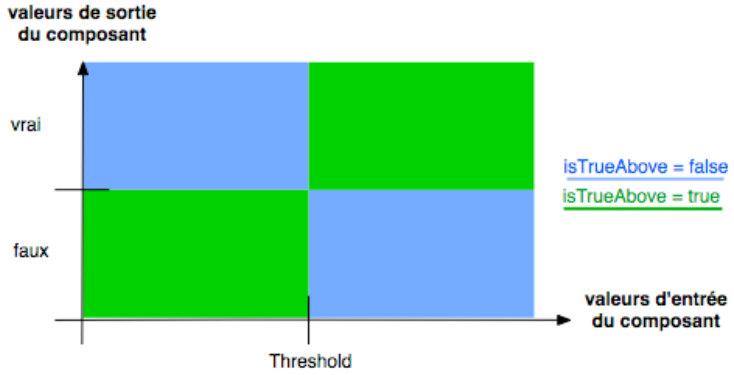


Figure 310 : Transformation effectuée par le composant *ThresholdInputLanguage*.

La Figure 206 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 206, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le paramètre *pThreshold* (Figure 206 ligne 2 et Figure 205 en abscisse) est un paramètre du constructeur qui permet de fixer le seuil.
3. Le paramètre *pIsTrueAbove* indique si la valeur de sortie est vraie au-dessus du seuil (Figure 205 en vert) ou au-dessous du seuil (Figure 205 en bleu).

```
1 | ThresholdInputLanguage(char* pName,
2 |                        int pThreshold,
3 |                        bool pIsTrueAbove);
```

Figure 311 : Pour créer un langage de liaison « seuil ».

Pour connecter ce composant, OP fournit un signal et un slot. Pour connecter un dispositif d'entrée à ce composant, alors le slot proposé est :

```
void update(int pMessage, QTime pTime);
```

Le premier paramètre est la valeur entière reçue en entrée, le second est une estampille temporelle.
Pour connecter ce composant à un autre composant, comme à une propriété numérique, les signaux proposés sont :

```
void updated(bool pMessage, QTime pTime);
void updated(QVariant pMessage, QTime pTime);
```

Pour les deux signaux le premier paramètre indique la valeur booléenne renvoyée par le composant, le second est une estampille temporelle.

3. DelayLanguage

Indifféremment composant d'une modalité de liaison en entrée ou en sortie (*pDirection* à la Figure 207), ce composant implémente une attente : il ne délivre en sortie sa valeur d'entrée qu'après un certain temps. Cette durée (*pDelay* à la Figure 207)) est une propriété du composant et est ajustable par l'utilisateur d'OP.
La Figure 207 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 207, ligne 1) et indique quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).

2. Le second paramètre, *pDirection* (Figure 207, ligne 2), indique si le composant fait partie d'un langage de liaison en sortie ou en entrée.
3. Le paramètre *pDelay* (Figure 207, ligne 3), indique le temps, en secondes, avant que la valeur d'entrée soit renvoyé en sortie.

```
1 | DelayLanguage(char* pName,
2 |               LinkingComponent::LinkingComponentDirection pDirection,
3 |               int pDelay);
```

Figure 312 : Pour créer un langage de liaison « délai ».

Pour connecter ce composant, OP fournit un signal et des slots. Pour l'entrée, les slots proposés sont :

```
void update(bool pMessage);
void update(bool pMessage, QTime pTime);
void update(QVariant pMessage, QTime pTime);
void update(QVariant pMessage);
```

Les premiers paramètres sont les valeurs reçues en entrée, le second (s'il est présent) est une estampille temporelle.
Pour connecter ce composant à un autre composant, alors le signal proposé est :

```
void updated(QVariant pMessage);
```

Le paramètre indique la valeur renvoyée par le composant.

4. RepeatLanguage

La Figure 208 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 208, ligne 1) et permet de fixer quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le second paramètre, *pDirection* (Figure 208, ligne 2), permet de fixer si le composant fait partie d'un langage de liaison en sortie ou en entrée.
3. Le paramètre *pRepeat* (Figure 208, ligne 3), permet de fixer le nombre de répétitions.
4. Le paramètre *pInterval* (Figure 208, ligne 4) permet de fixer la durée (en secondes) entre chaque répétition.

```
1 | RepeatLanguage(char* pName,
2 |                LinkingComponent::LinkingComponentDirection pDirection,
3 |                int pRepeat,
4 |                float pInterval);
```

Figure 313 : Pour créer un langage de liaison « répétition ».

Pour connecter ce composant, OP fournit un signal et des slots. Pour l'entrée, les slots proposés sont :

```
void update(bool pMessage, QTime pTime);
void update(QVariant pMessage, QTime pTime);
void update(QVariant pMessage);
```

Pour les trois signaux, le premier paramètre est la valeur reçue en entrée, le second (s'il est présent) est une estampille temporelle.
Pour connecter ce composant à un autre composant, le signal proposé est :

```
void updated(bool pMessage);
```

Le paramètre indique la valeur renvoyée par le composant.

5. ShortDisplayOutputLanguage

La Figure 314 présente le constructeur de ce composant :

1. Le premier paramètre du constructeur est *pName* (Figure 314, ligne 1) et permet de fixer quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée).
2. Le second paramètre, *pInterval* (Figure 314, ligne 2), permet de fixer le temps d'affichage.
3. Le paramètre *pType* (Figure 314, ligne 3), permet de fixer le type de ce qui sera affiché : il peut s'agir d'une image ou d'un message texte, par exemple.

```
1 | ShortDisplayOutputLanguage(char* pName,
2 |                            int pInterval,
3 |                            QVariant::Type pType);
```

Figure 314 : Pour créer un langage de liaison en sortie *ShortDisplay*.

Pour connecter ce composant, OP fournit des signaux et des slots. Pour l'entrée, les slots proposés sont :

```
void Show(QImage *pMessage);
void Show(QVariant *pMessage);
void Show(QVariant pMessage);
```

Le paramètre est la valeur reçue en entrée.
Pour connecter ce composant à un autre composant, le signal proposé est :

```
void IsShown(QImage *pMessage);
void IsShown(QVariant* pMessage);
```

Le paramètre indique ce qui a été affiché par le composant.

6. BeepOutputLanguage

La Figure 315 présente le constructeur du composant. Le premier paramètre du constructeur est *pName* (Figure 315, ligne 1) et permet de fixer quelle chaîne de caractères représente le nom du composant, tel qu'il est affiché dans l'interface graphique (si elle est utilisée). Le second paramètre, *pSoundFile* (Figure 315, ligne 2), permet d'indiquer quel fichier son sera joué. Si cette chaîne de caractères est laissée vide, alors aucun son ne sera joué. C'est le cas par exemple si le composant est connecté à une diode via un *PhidgetInterfaceKitDevice* (section III A 3 du Chapitre 6).

```
1 | BeepOutputLanguage(char* pName,
2 |                    const QString pSoundFile);
```

Figure 315 : Pour créer un langage *Beep*.

Pour connecter ce composant, OP fournit un signal et des slots. Pour l'entrée, les slots proposés sont :

```
void update(QVariant);
void update(bool);
void update();
```

Le paramètre, s'il est présent, ne l'est que pour permettre à ce slot d'être connecté à d'autres signaux, mais n'est pas utilisé.
Pour connecter ce composant à un autre composant, le signal proposé est :

```
void updated(bool pMessage);
```

Le paramètre indique si le composant a été activé ou vient de s'arrêter.

Annexe E : Questionnaire d'évaluation de OP

Tests sur un outil de conception

Prénom **date**

Quels sont les langages informatiques que vous utilisez le plus souvent ?

Quel est votre sujet de recherche ?

Quels sont vos sujets d'enseignement ?

Quels sont les types d'interfaces utilisateurs que vous développez habituellement?

☐ 1.ligne de commande ☐ 2.graphique ☐ 3.web ☐ 4.autre

Autres (précisez)

Avez-vous l'habitude d'utiliser des outils de conception pour le logiciel en général ?

○ 1.Oui ○ 2.Non ○ 3.Ne sait pas

si oui, lesquels

Avez-vous l'habitude d'utiliser des outils pour concevoir des IHM ?

○ 1.Oui ○ 2.Non ○ 3.Ne sait pas

si oui, lesquels

Avez-vous l'habitude d'utiliser des outils pour concevoir la partie logicielle des IHM ?

○ 1.Oui ○ 2.Non ○ 3.Ne sait pas

Si oui, lesquels

De manière générale, pour vous quelles sont les qualités d'un outil logiciel ?

Pouvez vous indiquer votre année de naissance ?

Vous êtes

○ 1.Une femme ○ 2.Un homme

Quel est le dernier diplome que vous avez obtenu ?

Tests sur un outil de conception

Questionnaire premier exercice - temps 1 -

Prénom | **date**

Avez vous déjà codé ce type de problème ?

○ 1.Oui, tous les jours ○ 2.Oui souvent ○ 3.Oui quelques fois ○ 4.Non ○ 5.Ne sais pas

Habituellement, comment faites vous ?

Dans l'idéal, comment aimeriez vous coder ce problème ?

Tests sur un outil de conception

Questionnaire premier exercice - temps 2-

Prénom **date**

Comment pouvez vous imaginer une architecture logicielle adaptée à ce type de développement ?

Pouvez vous en faire un schéma ?

date

Résumé

Depuis plusieurs décennies, la recherche en Interaction Homme-Machine travaille sur les systèmes qui mélangent le monde physique avec le monde numérique. Les interfaces de réalité mixte visent à faire disparaître la frontière entre ces deux mondes. Face au manque de capitalisation et d'aide lors de la conception de ces systèmes, nos travaux ont pour objectif de définir un modèle d'interaction avec les systèmes de réalité mixte unificateur capitalisant les approches du domaine existantes. Ce modèle, appelé Modèle d'Interaction Mixte, permet au concepteur de décrire, caractériser et explorer les techniques d'interaction. Pour cela, nous considérons les objets mixtes, à la fois physique et numérique, qui prennent part à l'interaction avec l'utilisateur.
Pour opérationaliser ce modèle d'interaction pendant la conception et permettre la réalisation d'ébauches interactives simultanément, nous avons développé un outil de prototypage. Cet outil reprend les concepts du Modèle d'Interaction Mixte.
Nous avons validé ces travaux en considérant deux formes de validations : conceptuelles et expérimentales. Nous avons considéré des cadres de validation existants et nous avons conçu plusieurs systèmes en collaboration avec des acteurs de la conception.

Mots-Clés

Interaction Homme-Machine, Realité Mixte, Conception, Prototypage.

Abstract

Research in Human-Computer Interaction explores Mixed Reality Systems for several decades. Mixed interfaces seek to smoothly merge the physical and digital (data processing) environments. Facing a lack of capitalization and help when designing such systems, our work aims at defining a unifying model of interaction with mixed reality systems, capitalizing existing approaches in this domain. This model, called Mixed Interaction Model, allows a designer to describe, characterize and explore interaction techniques. In order to do so, we focuse on physical-digital objects taking part in the interaction with the user.
In order to operationalize this interaction model during design and allow designers to realize interactive sketches simultaneously, we developped a prototyping tool. This tool capitalizes the concepts of the Mixed Interaction Model.
We conducted conceptual and experimental evaluations, by considering existing validation frameworks and we designed several systems in collaboration with design actors.

Keywords

Human-Computer Interaction, Mixed Reality, Design, Prototyping.