HAL
open science

Gestural interaction techniques for handheld devices combining accelerometers and multipoint touch screens

Adriano Scoditti

To cite this version:

Adriano Scoditti. Gestural interaction techniques for handheld devices combining accelerometers and multipoint touch screens. Other [cs.OH]. Université de Grenoble, 2011. English. NNT : 2011GRENM041 . tel-00665047

HAL Id: tel-00665047

https://theses.hal.science/tel-00665047v1

Submitted on 1 Feb 2012

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

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

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Adriano SCODITTI

Thèse dirigée par **Joelle COUTAZ**
et codirigée par **Renaud BLANCH**

préparée au sein **du Laboratoire d'Informatique de Grenoble équipe Ingénierie de l'Interaction Homme-Machine**
et de **l'École Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique**

Gestural interaction techniques for handheld devices combining accelerometers and multipoint touch screens

Thèse soutenue publiquement le **28 Septembre 2011**,
devant le jury composé de :

Michel BEAUDOUIN-LAFON

Professeur, Université Paris XI, Rapporteur

Marie-Paule CANI

Professeur, Grenoble I.N.P., Présidente

Eric LECOLINET

Maître de Conférences, Université Telecom-ParisTech, Examinateur

Franck POIRIER

Professeur, Université de Bretagne-Sud, Rapporteur

Nicolas ROUSSEL

Directeur de Recherche, INRIA Lille Nord-Europe, Examinateur

Abstract

In this thesis, I address the question of gestural interaction on mobile devices. These, now common, differ from conventional computers primarily by the input devices the user interact with (small screen size but tactile, various sensors such as accelerometers) as well as the context in which they are used. The work I present is an exploration of the vast area of interaction techniques on these mobile devices. I structure this space by focusing on the techniques based on accelerometers for which I propose a taxonomy. Its descriptive and discriminant power is validated by the classification of thirty-seven interaction techniques in the literature. The rest of my work focuses on the achievement of gestural interaction techniques for these mobile devices. With TouchOver, I show that it is possible to take advantage of complementary two-channel input (touch screen and accelerometers) to add a state to the finger-drag, thus enriching the interaction. Finally, I focus on mobile device menus and I propose a new form of sign language menus. I discuss their implementation with the GeLATI software library that allows their integration into a pre-existing GUI toolkit.

Acknowledgements

Thanks to everyone who though this thesis could be a reality. Thanks to everyone who contributed to this thesis and collaborated during my doctoral researches. Thanks to everyone who though my name could be compatible with the “doctor” prefix. Thanks to Joelle et Renaud, my advisors, who patiently listened to my initial hypotheses and supported them. Thanks to the jury and to their comments that contributed to improve the final version of this work and made the defense an unforgettable great experience. Thanks to Valentina, my mother, my brother and Raclet(te) who tolerated my volatile mood during the writing of this thesis.

Contents

I. Introduction 3

1. Introduction 5
1.1. Context and Motivations . . . 5
1.2. The Importance of Gestures . . . 6
1.3. A Palette of Gestural Interfaces . . . 7
1.3.1. Gestures in mid-air . . . 7
1.3.2. Touch-based gestures . . . 8
1.3.3. Manipulative gestures . . . 10
1.3.4. Multimodal Interaction . . . 10
1.4. Objectives of the work . . . 12
1.4.1. Gestural Interaction Techniques . . . 12
1.4.2. Gestures-Driven menus . . . 13
1.5. Summary of the contributions . . . 13
1.6. Outline of the dissertation . . . 14

II. Accelerometers-based gestural interaction techniques: Design Space 17

2. Classifying and Characterizing Gestures 19
2.1. The functions of gesture . . . 20
2.1.1. The function of gestures from a linguistic perspective . . . 20
2.1.2. The function of gestures in instrumental interaction . . . 22
2.2. Gestures morphology or Gesture styles . . . 25
2.2.1. The foundations from Quek: manipulative and semaphoric gestures 25
2.2.2. Karam's et al. gesture styles . . . 26
2.2.3. The case for gestural dynamics . . . 28

2.3. Gestures and enabling technologies 30
2.3.1. Gestures and enabling technologies according to Baudel 30
2.3.2. Gestural interaction techniques according to Roudaut and Hinckley 31
2.3.3. Gestures and enabling technologies according to Hinckley 32
2.4. The Need for a Unified View 34
2.5. Synthesis 38

3. A taxonomy for gestural Interaction Techniques based on accelerometers 39
3.1. The Foundations 40
3.1.1. Linguistics-inspired taxonomies 41
3.1.2. Morphological Taxonomies 44
3.2. A New Taxonomy for Accelerometer-based Gestural Interaction 46
3.2.1. Lexical Axis 48
3.2.2. Syntactic Axis 49
3.2.3. Semantic Axis 50
3.2.4. Pragmatic Axis 50
3.3. Classification of WIMP Techniques 51
3.4. Synthesis 52

4. Gestural Accelerometers-Driven techniques: a State of the Art 55
4.1. Application of the proposed taxonomy 55
4.2. Discussion 64
4.2.1. My Definitions 66
4.3. Synthesis 67

III. Composing Touch and Tilt 69

5. TouchOver 71
5.1. Introduction 71
5.1.1. The Need for Multiplexing Interaction Techniques 72
5.1.2. Precision of Selection 72
5.1.3. Introducing a Passive Tracking State 73
5.2. Observations 73
5.2.1. State Models of Input Devices 73
5.2.2. Improving Precision of Selection on Touch Devices 75
5.2.3. Multimodal Techniques 76

5.3. The TouchOver Technique . . . 77
5.4. Experimental evaluation . . . 79
5.4.1. Apparatus . . . 80
5.4.2. Participants . . . 80
5.4.3. Procedure . . . 80
5.4.4. Design . . . 81
5.4.5. Results . . . 82
5.5. Discussion . . . 87
5.6. Synthesis . . . 88

IV. Gestures-Driven Menus 89

6. Characterizing Gestural Menus 91
6.1. What is a Menu? . . . 92
6.1.1. Existing definitions for Menus . . . 92
6.1.2. Interaction Object, Widget, Interactor, Instrument, Interaction technique . . . 94
6.1.3. Synthesized Definitions/Choices . . . 97
6.2. Two taxonomies for Menus . . . 100
6.2.1. Shneiderman . . . 100
6.2.2. Bailly's MenUA design space . . . 104
6.3. Gestural Menus . . . 106
6.3.1. Kurtenbach's Marking Menus . . . 107
6.3.2. Bau's Octopocus . . . 108
6.3.3. Appert's gestural shortcuts . . . 110
6.3.4. Bailly's Flower and Leaf Menus . . . 111
6.3.5. Roudaut's Analysis . . . 113
6.4. Synthesis . . . 116

7. GeLATI: integrating hierarchical gestural menus in existing toolkits 119
7.1. Existing approaches . . . 120
7.1.1. Rubine's GRANMA . . . 120
7.1.2. Wobbrock's $1 . . . 122
7.2. Objectives and Approach . . . 124
7.3. GeLATI templates . . . 125
7.3.1. Structure, Aspect and Interaction . . . 127

7.3.2. Software Architecture . . . 129
7.4. Examples of Use . . . 132
7.4.1. Implementing reference menus with GeLATI . . . 132
7.4.2. Integrating GeLATI in legacy toolkits . . . 135
7.4.3. Multimodality with GeLATI . . . 136
7.4.4. Parallel interaction with GeLATI . . . 137
7.5. Limitations . . . 137
7.5.1. GeLATI intrinsic limitations . . . 138
7.5.2. Future work . . . 138
7.6. Synthesis . . . 139

V. Conclusions **141**

8. Conclusions **143**
8.1. Contributions . . . 143
8.1.1. Gestural Classification . . . 144
8.1.2. GeLATI . . . 145
8.1.3. TouchOver . . . 146
8.2. Limitation and Perspectives . . . 146
8.2.1. The re-selection mechanism . . . 147
8.2.2. Non-rectilinear traits . . . 147
8.3. I should have... . . . 147

A. Technical Annex **195**
A.1. API . . . 195
A.1.1. UMMenu header . . . 196
A.1.2. QuestionMark View Controller . . . 197
A.2. XML Files . . . 199
A.2.1. QuestionMark GeLATI Definition: Structure.plist . . . 199
A.2.2. QuestionMark GeLATI Definition: Interaction.plist . . . 200
A.2.3. QuestionMark GeLATI Definition: Aspect.plist . . . 202

List of Figures **209**

List of Tables **217**

Bibliography **219**

Part I.

Introduction

Chapter 1.

Introduction

1.1. Context and Motivations

This thesis is concerned with the design and development of gesture-based interaction techniques for handheld devices. Handheld and mobile devices are often presented as desktop computers with limited computational capabilities and restricted input/output bandwidth and comfort. In this thesis, I start from the alternate perspective that mobile devices are fundamementally different from desktop computers, and therefore the desktop metaphor is inappropriate for mobile devices. This point of view is motivated by two observations. First, desktop computers and handheld devices are solutions for different contexts of use. Second, they offer very different fundamental features for interaction.

Smartphones such as those presented in Figure 1.1, audio players and tablet PCs have already reached high performance while preserving relatively small and compact size. Their compactness allows users to perform their tasks while on the move without the constraints of desktop PCs. Checking e-mail or browsing favorite websites while on the bus, listening to music while running, or keeping trace of appointments whenever and wherever needed, are typical examples of everyday life scenarios.

By means of inexpensive sensor technologies embedded into mobile devices, human skills and abilities can now be capitalized in novel ways. In particular, a whole new range of opportunities for physical interaction based on human manipulative skills has opened. Instead of interacting with computers *through* physical devices such as mice and keyboards, interaction can occur *with* the mobile device itself. Perhaps, the best illustration of this trend is the concept of "embodied user interfaces" [Fishkin 98] or that

Figure 1.1.: From the left: a Microsoft Windows Mobile 7 interface screenshot showing a perspective animation; the Apple iPhone home screen; the Palm Pre task selection interface.

of "manipulative user interfaces" [Harrison 98] which, in turn, are specific approaches to "Tangible User Interfaces" [Fitzmaurice 93, Ishii 97].

1.2. The Importance of Gestures

Gestures are woven inextricably into our lives. Quoting Axtell: "Without gestures, our world would be static, colorless... Mario Pei, a communication expert, once estimated that humans can produce up to 700,000 different physical signs. Birdwhistell estimates that the face alone is capable of producing 250,000 expressions and reports that researcher M. H. Krout identified 5,000 distinct hand gestures" [Axtell 91]. These numerical findings indicate that gestures can play a significant role in Human Computer Interaction (HCI). They also express the tremendous difficulty for HCI researchers to design and develop gesture-based interaction techniques that are effective.

According to Baudel, gesture-based interaction techniques are effective in that they can be concise, natural, and direct [Baudel 95, Norman 10, Morrel-Samuels 90]:

Concise when they permit users to specify both a command and its parameters as an atomic action.

Natural when they match users expectation with respect to the command they are invoking.

Direct when they make direct manipulation really direct, not through the indirection of physical input devices.

On the other hand, "gestural systems are no different from any other form of interaction" [Norman 10] in that they need to follow the basic rules of interaction design: Typically, an appropriate conceptual model must be devised, provision for feedforward and feedback, as well as mechanisms for interruptability and undo-ability must be of major concerns. In addition, because of their naturalness, gestures may be ambiguous and may be addressed unintentionally at the system level.

Although gestural interaction has been studied since the early sixties, we still do not have standard conventions of the same nature as the interaction patterns that have been developed for WIMP user interfaces. As shown by the examples below, gestural interaction has been tackled from many prospective directions giving rise to a prolific number of solutions.

1.3. A Palette of Gestural Interfaces

The following examples illustrate the breadth of the current solutions. They are not intended to provide a complete overview of the state of art. A more detailed analysis of the state of the art will be presented in Chapter 2. We roughly observe four groups of gestural interaction techniques: gestures in mid-air, touch-based gestures on surfaces, manipulative gestures, and gestures combined with other modalities.

1.3.1. Gestures in mid-air

Myron Krueger's pioneering work on artificial reality in the early 1980s is perhaps the first introduction to body gestural interaction in mid-air with large, projected images using a video-camera for full-body tracking.

Baudel's Charade is the first French illustration of 3D hand gestures using a DataGlove. Figure 1.3 shows the configuration of the Charade interaction system as well as an example of the gestural vocabulary to control a slideshow application.

Figure 1.2.: Krueger is one of the pioneer researchers of full body gesturing.

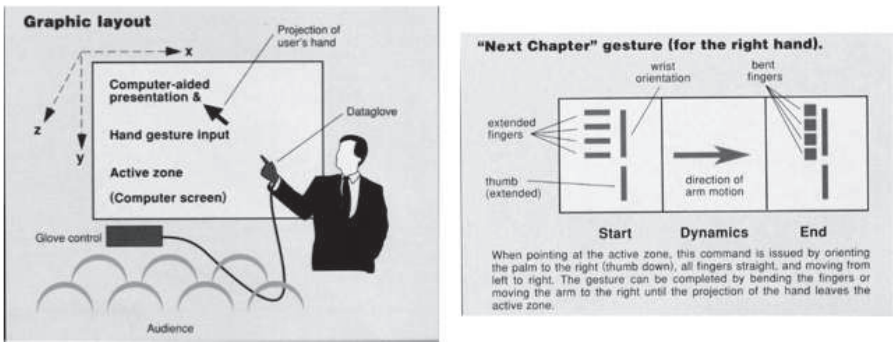


Figure 1.3.: Charade uses a 3D hand gestures interaction model. On the right, an example of the notation used to denote the "next chapter" gesture [Baudel 95].

1.3.2. Touch-based gestures

The number of touch-based gestures on graphic tactile surfaces is literally exploding. As an answer to this diversity, Villamore *et al.* have created a Touch Gestures Reference Guide to organize and classify all possible gestures and their availability on current commercial products [Villamore 10]. In addition, Wobbrock *et al.* analyzed the usability of user defined tabletop gestures [Wobbrock 09].

Among these systems, Bau's OctoPocus [Bau 10] and Roudaut's MicroRolls [Roudaut 10] deserve particular attention. OctoPocus combines immediate feedback with feedforward in a tightly manner as the user produces the gesture. By showing all possible paths incrementally, (1) users know what they have just done (feedback) and where they are

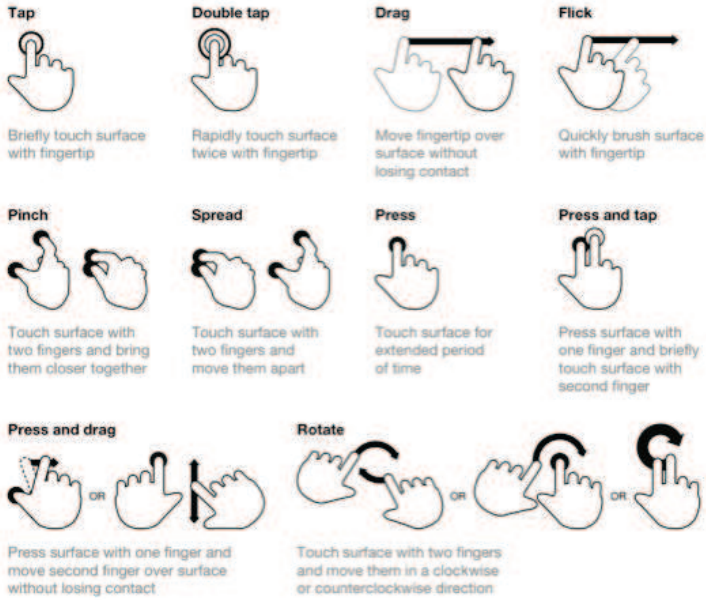


Figure 1.4.: Villamore *et al.* propose a reference guide for all touch-based gestures implemented in current modern systems such as iOS, Windows Mobile 7 or WebOS-based mobile devices [Villamore 10].

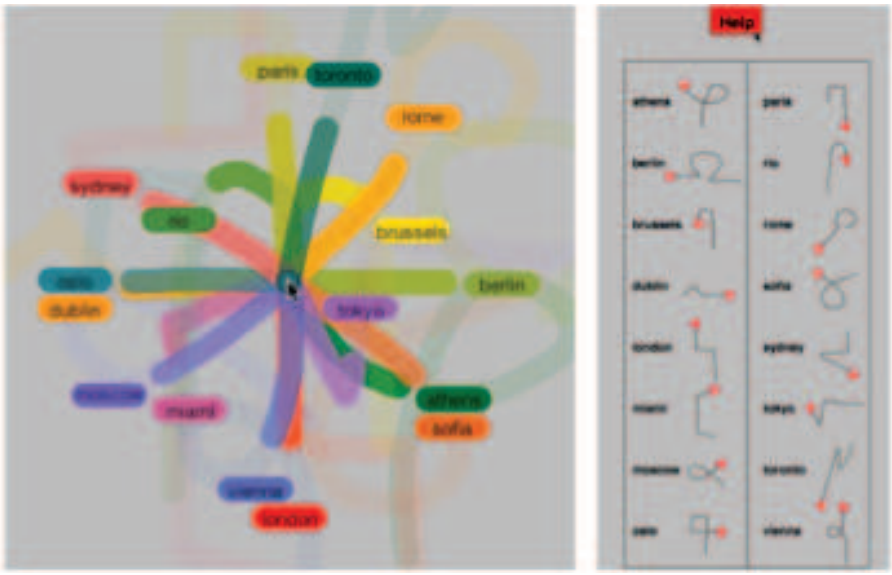


Figure 1.5.: OctoPocus integrates feedback with feedforward in gesture-driven interfaces to help users to discover and learn gestural menus [Bau 10].

going (feedforward), and (2) both experts and novices are supported in a flexible manner. Figure 1.5 shows an example of an OctoPocus menu along with the gestural vocabulary.

The MicroRolls interaction technique exploits thumb micro-gestures as a mechanism to enrich input vocabulary while requiring small "foot-print" on the screen real estate.

Figure 1.6.: Roudaut proposes the exploitation of micro-gestures on touch screen based devices with "Microrolls" [Roudaut 10].

As shown in Figure 1.6, micro-gestures are accomplished by leaning the finger in six different orientations without the need to translate the finger on the screen. Six different commands are associated with the six distinct MicroRolls. Additionally, MicroRolls can be combined with finger translation to propose a seamless touch driven gestural interaction as proposed in the RollMark menu concept Figure 1.6.

1.3.3. Manipulative gestures

Manipulative gestures such as squeeze, tilt, and shake, are applied to the physical body of the handheld device. With manipulative gestures, *the body of the device is part of the user interface*, thus, the term "embodied user interface" [Fishkin 98]. As an example of manipulative gesture (see Figure 1.7), the user tilts the device to switch among previously opened applications activating the modality by a simple "tap" gesture on the back of the device. Other typical (and pioneering work in this area) include [Fitzmaurice 93, Hinckley 00, Levin 99, Partridge 02, Rekimoto 96] who has paved the way for an active research area [Ballagas 06, Williamson 07, Wilson 03].

Gesture is also an input modality, which as such, can be combined with other modalities such as speech.

1.3.4. Multimodal Interaction

Bolt's "put-that-there" serves as the paradigmatic reference for multimodal interaction where speech and gesture can be used in a complementary way to manipulate graphics shapes in mid-air (see Figure 1.8). Typically, gesture is used as deictics as in: "Move <u>that</u> to the right of the green square" or "Put <u>that</u> <u>there</u>".

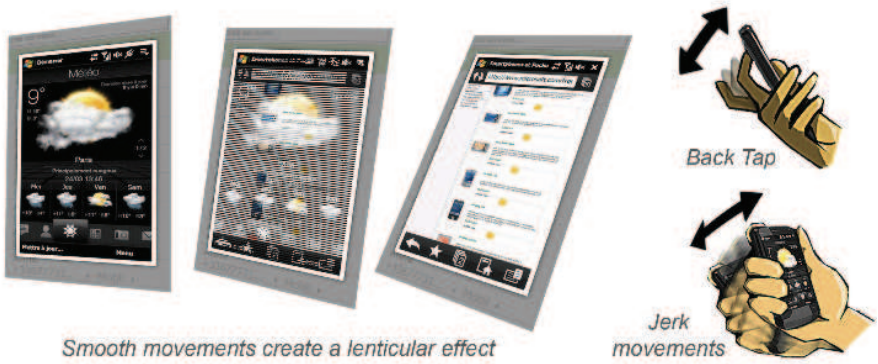


Figure 1.7.: TimeTilt proposes an example of multimodal interaction that exploits both accelerometers coupled with two different interaction languages [Roudaut 10].

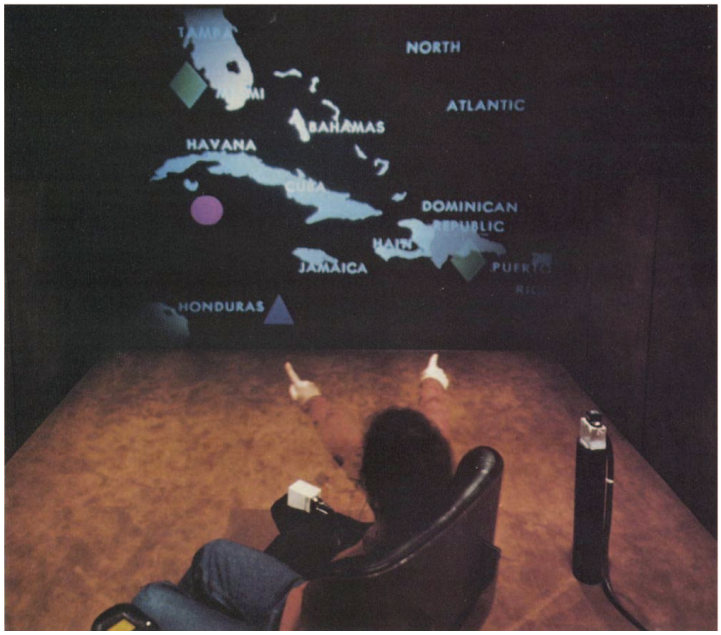


Figure 1.8.: Bolt used the combination of voice and hand gestures to command the "Media Room" to controll shape based application [Bolt 80].

Multimodal interaction based on speech and deictic gestures has been studied since the early eighties. A good number of solutions and toolkits are now available where multimodal interaction is supported for speech, pen-based gesture, and for mid-air gestures [Oviatt 92, Cohen 89, Chatty 04]. On the other hand, the problem of multimodal interaction involving accelerometers as input devices has not been addressed in a comprehensive manner. Sensors alone are not always capable to fully determine whether an interaction has started, continued or ended. A complementary modality usually offers a natural way for the user to denote these markers. For example, Hinckley *et al.* have explored the synergistic complementarity of touch and motion for hand-held devices [Hinckley 11].

In addition, native operating systems and toolkits for handheld devices have not been designed to support the integration of novel interaction techniques. In particular, the integration of new techniques to control preexisting widgets is not trivial. In most cases, the underlying architecture is strictly connected to the WIMP interaction properties resulting in strong technical limitations. For example, since a mouse pointer has always a position, graphical users interfaces widgets behavior are often driven by the pointer position. Such a small detail becomes an enormous limitation when trying to control the same widget using an accelerometers-driven modality.

This work have two main objectives: (1) the organization of gestures-based menus in particular those exploiting multimodal inputs (touch screens and accelerometers); (2) the proposition of new gestural interaction techniques based on the integration of touch and accelerometers input devices.

1.4. Objectives of the work

1.4.1. Gestural Interaction Techniques

Different taxonomies characterizing gestural interactions have already been proposed in the Human-Computer Interaction (HCI) literature. In this thesis, my first goal is to propose a state of the art of existing approaches and a categorization of existing techniques. The synthesis of existing approaches will permit the enhancement of them through the proposition of a user centered taxonomy thanks to which I will be able to organize the state of the art in accelerometers-driven user interaction techniques.

My second goal is to achieve a seamless interaction enabling both touch screens and accelerometers input devices so that they work together both in collaboration and as alternatives (Complementary and Redundancy in terms of the CARE properties [Coutaz 95]) to propose a multimodal interaction.

1.4.2. Gestures-Driven menus

Gestural driven interaction techniques are often used to control graphical menus. In order to offer a rich user experience, I need to overcome the specific difficulties of gesture-based interactions. As described by Baudel [Baudel 95], those difficulties still need a clear solution and a proper architectural integration:

Interpretation Algorithm. I want to find a generic approach to offer an algorithm able to interpret differently shaped gestures. I do not want to limit the algorithm with a learning phase.

Gestural Characteristics. I need to understand which class of gestures is best suitable on mobile devices and touch/accelerometers input characteristics.

Chosen Interaction. Existing works on gesture based interaction techniques present two main characteristics: (1) an off-line interpretation of interactions and (2) a single hierarchy level in gestures organization. I want to handle on-line tracking and gesture recognition and the ability to organize gestures in menus, thus proposing personalizing menu structures and hierarchical levels.

1.5. Summary of the contributions

I propose an overview of existing approaches to gestural interaction techniques. Different works from the past 20 years will be organized in order to focus on communication capabilities of gestural interactions in the field of HCI. I propose an updated state of the art in existing accelerometers-driven gestural interaction techniques for mobile devices. Those techniques are organized thanks to a novel user centered taxonomy. I present TouchOver, a new interaction techniques to perform a selection task on accelerometers- and touch-enabled mobile devices. My last contribution is the GeLATI gesture recognition library. GeLATI implements a vectorial driven recognition algorithm for rectilinear gestures composing gestural menus.

1.6. Outline of the dissertation

This thesis is structured into five main parts, the first one being this introduction and the last one the final conclusions. The central parts compose the kernel of this work and are described above in details:

Part II Describes the scientific and industrial state of the art of accelerometers-based interaction techniques on mobile devices. In particular:

1. The first chapter studies the characterization of gestures according to existing taxonomies and definitions. Several works are presented and the scope of this work is defined by the gesture classes I will address.
2. I present the foundations for taxonomies of interaction techniques and input devices. I present a new taxonomy for gestural interaction techniques. I classify some well-known WIMP interaction techniques to best clarify the use of the proposed taxonomy.
3. I then use this new taxonomy to classify the state of the art of accelerometer-driven interaction techniques. I describe over twenty interaction techniques that have been proposed during the last twenty years. I conclude by broader discussion about the general frame proposed by the classification and how it helped in finding the path of my research.

Part III My technical contributions consist in the proposition of TouchOver, a selection interaction techniques and a gestural infrastructure to implement multimodal interaction techniques. This part of the work introduces TouchOver:

1. I propose TouchOver, a multi-modal interaction technique, that decouples positioning and selection elementary tasks on touchscreen- and accelerometer-enabled handheld devices. With TouchOver, positioning is performed with a finger on the touch surface, while selection is conveyed by a gentle tilt of the device. By doing so, TouchOver adds a mouseover-like state and improves selection precision while remaining compatible with existing precision-improvement interaction techniques.

Part IV Shows that a set of gestures organized in a graphical widget can compose a menu. Those graphical gestural menus need to be well integrated with existing widgets while exploiting the multiple input characteristics offered by the device:

1. I present a concise state of the art of graphical menus. I analyze existing definitions and examples in order to frame the domain.
2. This section is dedicated to the third contribution of this work: the GeLATI library/architecture. GeLATI is a vectorial approach to rectilinear gestures recognition. Complex gestures are decomposed in traits. Traits are disposed in hierarchy in order to offer the opportunity to create gestural menus.

Part II.

Accelerometers-based gestural interaction techniques: Design Space

Chapter 2.

Classifying and Characterizing Gestures

In this chapter, I provide a synthesis of the taxonomies that have been developed for gestures. The objective is to propose a unified concise view for a complex prolific field so that a researcher, new in the field, can rapidly relate the different approaches, perspectives, and terminologies. Some taxonomies focus on the functions of gesture, others classify gestures according to their morphology while others blur the distinction between function and form.

Among the many functions of gesture, I am concerned with the role of arm-hand-finger gestures as a means for interacting with a computer system. The consequent question relates to the forms that support this role in an effective manner from both the human and system perspectives. For this purpose, I have used Karam's *et al.* taxonomy as a basis [Karam 05]. The first contribution from their work is that it results from a literature review of over 40 years of gesture based interaction; second, it is an attempt at proposing a unifying terminology. I have then extended or related Karam's taxonomy with more specific taxonomies such as that of Cadoz for instrumental gestures [Cadoz 94], or that of Roudaut and Baglioni for handheld devices [Baglioni 09]. The result of my synthesis is shown in Figure 2.6 and Tables 2.7 and 2.8 at the end of the chapter.

This chapter is organized according to the following sections:

The functions of gesture introduces gestures characterized according to their communication characteristics.

Gestures morphology or Gesture styles organizes gestures from a morphological point of view.

Gestures and enabling technologies presents a third point of view that characterizes gestures according to the technologies involved in the implementation of gestures acquisition and recognition.

Finally a unified point of view will compare the proposed approaches and clarify the adopted vocabulary.

2.1. The functions of gesture

Psycholinguists have been debating for years about the functions of gestures. On one side, gesture is seen as communicative. In this case, gesture is considered as a means to assist the listener. On the other side of the debate, gesture is seen as a means to assist the speaker in producing speech. In this case, gesture is derivative from the production of speech. Others such as Bavelas *et al.* cited in [Loehr 04], refer to interactive gestures as a subclass of communicative gestures whose role is "to help maintain the conversation as a social system". An interesting synthesis of the pros and cons of these theories can be found in [Loehr 04]. These studies have focused on the role of gesture in relation to speech in human-to-human communication. For the purpose of my doctoral research, I have selected two representative analyses of the functional roles of gestures: (1) Ekman's work, a proponent of "gesture as communicative" since I am concerned with the problem of commanding a system through gestures. (2) Cadoz's work who, contrary to the linguistic approach, does not consider speech, but instead the use of instruments as a basis for communication.

2.1.1. The function of gestures from a linguistic perspective

By the end of the sixties, Ekman and Friesen have proposed to organize gestures into five major functional categories: emblems, illustrators, affect displays, regulators, and adaptors (later renamed manipulators by Ekman himself) [Ekman 69].

Emblems such as the "OK" sign in Italian culture, are utilized for communication when verbal exchange is inhibited, typically by noise or distance. They do not require the presence of another modality such as speech, to insure their role. They are self-contained. On the other hand, they are culture-specific, therefore acquired through learning.

Illustrators refer to "movements which are intimately tied to the content and/or flow of speech" [Ekman 69]. As nonverbal behavior, their role is to emphasize, punctuate, or complete a verbal utterance. In his 1980 article, Ekman proposes subcategories for illustrators, most of them inspired from Efron's seminal work on gestures published in 1941 [Efron 41]. These include *deictics* to denote an entity (an object, idea, location); *Batons* (or beats) which are rhythmic gestures to punctuate a sentence and *rythmics* to depict the rythm or pacing of event; *Underliners* to emphasize a particular word or a group of sentences; *Ideographs*, such as traces or sketches out in the air, to portray the course of thought; *Spatials* that depict a spatial relationship; *Pictographs* where movements draw the shape of the referent in the air; *Kinetographs* where movement imitates a bodily action. For example, while pronouncing the words "so I finally wrote to him", the speaker uses the index finger of one hand to write upon the other hand ([Efron 41], p. 125 cited by [Loehr 04]). This long, although incomplete, enumeration illustrates the richness and terminological diversity in this area of research as well as the thin distinction made between forms and roles. Some of these terms will be reused later is Section 2.2 on gestures morphology.

Affect or emotional displays indicate momentary psychological states, affects and moods in reaction to stimuli. They result primarily from movements of facial muscles. Their role is mostly informative (rather than communicative) since people are generally not intentional in performing these behaviors.

Regulators "maintain and regulate the back-and-forth nature of speaking and listening between two or more interactants" [Ekman 69]. They support the interaction between sender and recipient. Common examples of regulators include head nods, eye contacts, and postural shifts. Regulators are distinguished from illustrators in that they direct and regulate the flow and pace of a conversation. Like affect displays, they are usually performed involuntarily. Therefore, there are mostly informative just like the immediate classic feedback in WIMP user interfaces, but according to Bavelas terminology referenced above, they are classified as interactive!

Adaptors (renamed later manipulators) refer to movements that involve manipulation of, or simply contact with some physical thing. It is an *object-adaptor* if the "thing" is a physical object (such as flipping a pen during exams!). It may be a *self-adaptor* when movements apply to a part of one's own body (e.g., scratching oneself, rubbing the eyes). It may be an *alter-adaptor* when it has to do with

interpersonal contacts (as in shaking hands to show polite and warm welcome to a visitor)[1].

Ekman's (and others) linguistic approach to gestural functions focuses on the relation of gesture with speech for interpersonal communication. Adaptors (manipulators) movements have been studied in cognitive activities in relation to stress or engagement, but not for a very different role, that is, as an instrument to produce, transform, and build artifacts, that is, as manipulators of tools (such as hammers and handheld devices) to change the state of the world. This dimension is well described by Cadoz for what he calls "instrumental interaction".

2.1.2. The function of gestures in instrumental interaction

Cadoz defines three complementary and imbricated functions for hand-based gestures: the epistemic, the ergotic, and the semiotic functions [Cadoz 94].

The epistemic function (from Greek epstēmē, knowledge) corresponds to the tactile and tactilo-kinesthesic (or haptic) capabilities of the hand. By moving hands and fingers on the surface of an object, people can rapidly get information about the shape, orientation, size of this object and from there, recognize and identify the object, thus the term "epistemic". This function is about exploring and acquiring knowledge about the physical world through dynamic touching with the body, but the hand, whose internal part has many receptors, plays a central role. In HCI, the epistemic function of the hand has motivated many forms of tactile feedback ranging from virtual reality applications to handheld devices (e.g., vibrators and more advanced ones such as that described in [Taylor 08]).

The ergotic function (from Greek ergon, work, force) is supported by the capacity of the hand to produce energy and from there to transform the state of the world. The hand can carry, assemble, reshape, and break things. Being involved in actions, the hand exchanges energy with the physical world which in turn reacts in many ways including sending energy back, thus the term "ergotic". In HCI, the ergotic function of the hand is at the foundation of direct manipulation user interfaces followed by graspable and embodied user interfaces, and more generally by tangible and reality-based interaction [Jacob 08].

[1] Interpersonal contacts form a continuum of intimacy levels which has set the foundations for proxemic interaction .

The semiotic function (from Greek sēmeion, sign) is about conveying meaning to third party through gestures. As presented above, this aspect has been widely studied in psycholinguistics for human-to-human communication. Reusing Ekman's *et al.* classification and definitions, emblems, illustrators and alter-adaptors are semiotic gestures. On the other hand, regulators and emotional displays are not semiotic since they are not produced consciously to convey meaning.

By making explicit three different roles for hand gesture, Cadoz is able to clarify the distinction between "*free-hand gesture*" ("*geste à nu*") whose role is purely semiotic, from "*instrumental gesture*" ("*geste instrumental*") which combines the epistemic, ergotic, and semiotic functions. As a communication means, instrumental gestures can be characterized in the following way:

1. Gestures are applied to an instrument, that is, to a physical object held in the hand, and there is interaction between the instrument and the person (epistemic and ergotic functions). The role of the instrument is to transform the energy produced by the hand into phenomena that can be perceived by a recipient (say another person, or a computer).

2. During the interaction between the instrument and the person, physical phenomena are produced and exchanged between the instrument and the person who, in turn, can dynamically modify the gesture in a fine grained manner to modulate the phenomena.

3. These phenomena produce information that is intended to make sense for the recipient (semiotic function).

Interestingly, Cadoz observes that an instrumental gesture is not monolithic, but structured into three components: (1) The **excitation gesture** provides the energy to produce the physical phenomenon; (2) The **modulation gesture** defines the permanence and/or the variations of the excitation gesture for the duration of the instrumental gesture. Modulation is further redefined in (2a) **continuous modulation** and (2b) **discrete modulation**; (3) The **selection gesture** or **deictic gesture** is the gesture component that chooses a particular component of the instrument (an instrument is rarely monolithic e.g., choose the cord of a guitar, or the appropriate mouse button of a mouse). Using the tilt gesture as an example of instrumental gesture, tilting the smartphone from the rest position corresponds to the excitation gesture, adjusting the tilting angle to control the speed of scrolling is a continuous modulation gesture while the selection gesture is

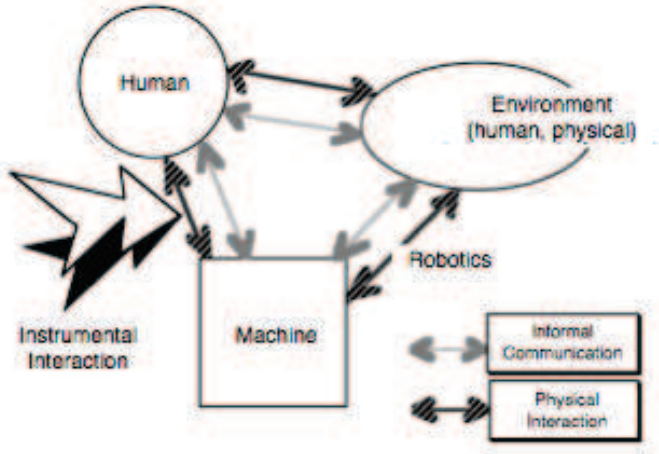


Figure 2.1.: Instrumental Interaction provide Humans with a means to interact with the Environment through the Machine [Cadoz 94].

mapped to positioning the device back to the rest angle or to any other means to denote the end of the instrumental gesture.

In [Cadoz 94], Cadoz extends the notion of instrument to any object with which humans can interact using their sensori-motor capabilities to accomplish a task. In particular, he uses the term *virtual instruments* to denote those objects that are digital. He states as a principle that for any task (or subtask), it is possible to define an instrument so that the interaction phenomena between the user and this object conveys the information that is necessary to accomplish the task. Then, instrumental gesture is an interaction modality that supports communication between humans and machines. This is illustrated by Figure 2.1. Figure 2.2 classifies gestures in a two-dimension space: whether gesture is ergotic or not, and whether gesture is communicative (e.g., between a human and a machine) or not. Within this space, instrumental gesture is necessarily ergotic: it implies the production of energy. Instrumental communication requires ergotic gesture that conveys meaning. In Chapter 6, I will relate Cadoz's generalized concept of *instrumental communication* to that of *instrumental interaction* introduced later by Beaudouin-Lafon [Beaudouin-Lafon 00]. Within the classification space of Figure 2.2, Ekman's regulators and affect displays lie at the bottom right quarter: they are non-ergotic and non communicative. Emblems and illustrators are communicative but not ergotic whereas self-adaptors are examples of non communicative but ergotic gestures.

In synthesis, Cadoz and Ekman provide two complementary perspectives for understanding the functional dimensions of gesture. In addition, Cadoz has opened the way

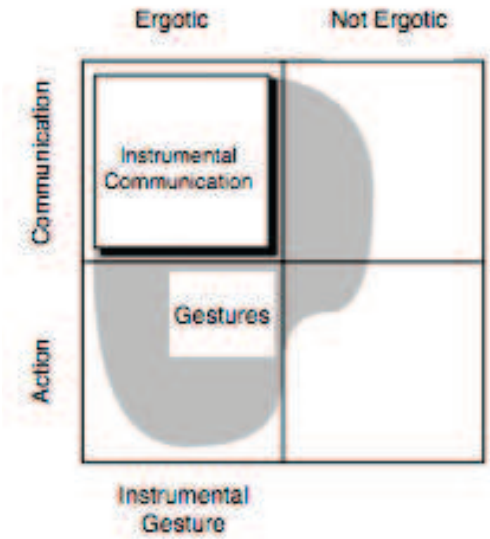


Figure 2.2.: Gesture in human computer interaction [Cadoz 94]. Instrumental gesture is primarily ergotic (it produces energy). It may be used for communication (it is then semiotic) or for pure action on the world (absence of meaning) [Cadoz 94].

to the analysis of gesture morphology with a three-components structure (excitation, modulation, and selection) but limited to instrumental gestures. In this regard, Karam's classification, presented next, has a wider coverage.

2.2. Gestures morphology or Gesture styles

Karam's *et al.* classification is an extension of that of Quek *et al.*, which in turn simplifies and, at the same time, augments the linguistic perspective of Ekman presented above.

2.2.1. The foundations from Quek: manipulative and semaphoric gestures

Quek *et al.* organize the major gesture styles developed in human-computer interaction into three classes: manipulative gestures, semaphoric gestures and gestures used in conjunction with speech [Quek 02]:

Manipulative gestures are gestures *whose intended purpose is to control some entity by applying a tight relationship between the actual movements of the gesturing hand/arm with the entity being manipulated.* Clearly, this definition is closely related to Cadoz's

concept of instrumental gesture although it does not refer explicitly to the presence of an instrument, whether it be physical or digital.

Semaphoric gestures are *any gesturing system that employs a stylized dictionary of static or dynamic hand or arm gestures.* In other words, these gestures rely on, possibly linguistic, conventions.

Gesture-speech corresponds to those gestures used with the speech modality *co-expressively.*

Semaphoric and gesture-speech gestures abstract away Ekman's emblems and illustrators functional categories presented above. If we refer to the two major metaphors for HCI described by Hutchin's *et al.* in [Hutchins 85, Norman 86b], manipulative gestures are ascribable to the real world metaphor where the user has the sensation of direct engagement (*le faire*), whereas semaphoric gestures are involved in the conversation metaphor where the user describes the desired state to the system (*le faire-faire*).

In addition, Quek *et al.* have observed that manipulative and semaphoric gestures differ in many ways. Their dynamics is different, their need for feedback is different, and their requirement for formality is different. Manipulative gestures require (or are aided by) tightly-coupled feedback from the object being manipulated which is not the case for semaphoric and gesture-speech gestures. This is another way of referring to the epistemic function of instrumental gestures that guides the ergotic modulation function in a tightly coupled manner. Semaphoric gestures, which are codified, impose various degrees of linguistic formalism on users. On the other hand, Quek's *et al.* classification does not leave any room for free-form gestures, also called gesticulation or natutal gestures. This is where Karam proposes a useful extension along with an attempt to standardize the terminology.

2.2.2. Karam's et al. gesture styles

As shown in Figure 2.3, Karam's *et al.* taxonomy is organized as a four dimension space [Karam 05]: gesture styles (morphology), system response (i.e. the system output modality used for feedback), enabling technology (whether gesture is acquired through physical input devices that require physical contact with the human body/hand or not), and the application domains. In the context of this section, we are concerned with the gesture styles axis where Quek's manipulative and semaphoric gestures are extended with gesticulation, deictics and sign languages.

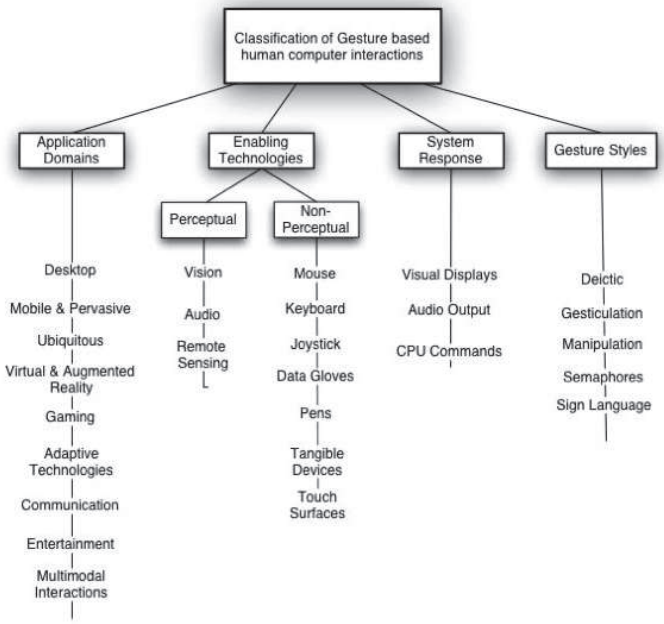


Figure 2.3.: The gestural interaction framework proposed by Karam *et al.* [Karam 05] enables a complete discussion about gestures themselves and the contextual variables/parameters associated with.

Karam *et al.* reuse the definition provided by Quek for deictic, manipulative and semaphoric gestures with some additional refinements. With the introduction of gesticulation and sign languages, they show some consideration for the continuum of increasing formality identified by McNeill in 1992 [McNeill 92]:

Manipulative gestures are refined by Karam according to the number of degrees of freedom (e.g., 2D and 3D), as well as according to the nature of the object manipulated (virtual or physical) and the presence or absence of a physical instrument (in Cadoz's sense). Manipulative gestures are thus sub-classified into four main classes: (1) Gesturing in two degrees of freedom for two-dimensional interactions; (2) Gesturing in multiple degrees of freedom for two-dimensional interactions; (3) Gesturing with tangible objects for three-dimensional interactions; (4) Gestures for real-world physical object interactions. Note that nothing is said about manipulative gestures for volumetric surfaces [Roudaut 11, Benko 08, Grossman 04].

Semaphoric gestures are codified (as in Quek's definitions): they rely on shared conventions between the interactants. In addition, Karam characterizes them with two other dimensions: whether they involve dynamic movements or static poses, and whether they are strokes (marks) or not. In turn, Bragdon *et al.* observe two types of

marks whose nature has an impact on human performance in mobile environments: those that are comprised of rectilinear segments (as exemplified by the marking menus), and those that are free-form paths (such as the pigtail) [Bragdon 11]. Interestingly, in their experiment, the authors found that rectilinear marks "*were faster and more accurate to perform, and were preferred by users to free-form paths*". This result will be exploited in my own work for the GeLATI library Chapter 7.

Deictic gestures have their own place, sitting between pure manipulation and pure natural gesticulation.

With regard to codification, McNeill makes a distinction between natural (free-form) gesticulation, pantomine, emblems, and sign languages [McNeill 92]:

Gesticulations and pantomimes are not codified. They have no linguistic properties in that there is no lexicon of agreed-upon symbols, and no phonological, morphological, and syntactic rules for combining these symbols. Both gesticulations and pantomines are *global gestures*: the meaning of the parts, if these parts exist, is determined from the meaning of the whole, top-down. In addition, gesticulation is *synthetic* (the meaning is conveyed within a single symbol) whereas pantomine is *analytic* (the meaning is conveyed across multiple symbols).

Emblems are partly codified, sign languages are fully conventionalized. Emblems are culture-specific, emblems and sign languages of them are *segmented* (the meaning of the whole is determined by the meaning of the parts, bottom-up), but emblems are generally *synthetic* while sign languages are *analytic*.

Although Karam's *et al.* classification mentions the dynamics for gestures, it only does so for semaphoric gestures.

2.2.3. The case for gestural dynamics

As stressed by Cadoz for instrumental interaction, dynamics is an important morphological feature that prevails in the excitation and modulation components. Dymanics is also mentioned in the definition of the functional roles of batons and rhythmics in human-to-human communication (cf. Section 2.3 above).

Baglioni, who has adapted Karam's taxonomy to handheld devices, introduces two axes for this purpose [Baglioni 09]: control and movement types as visible in Figure 2.4.

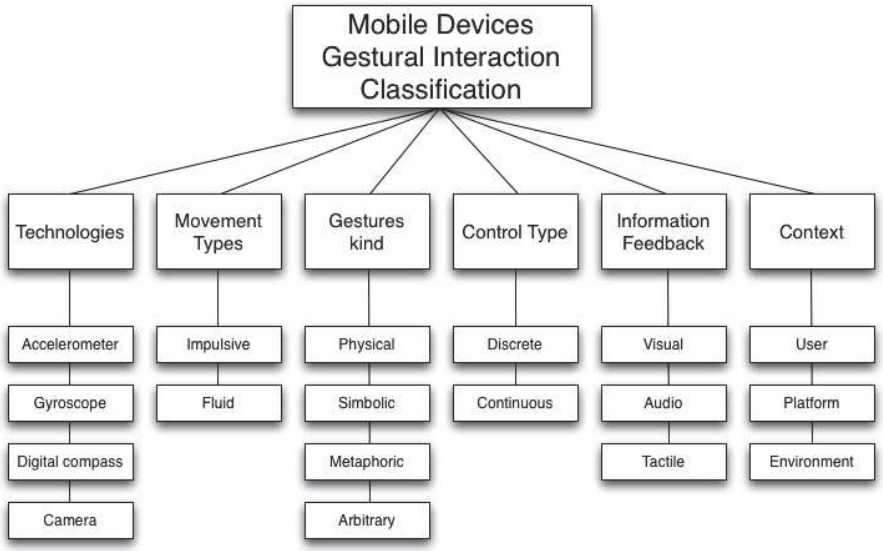


Figure 2.4.: Baglioni's characterization space enhance Karam's taxonomy to adapt to handheld devices. Adapted from [Baglioni 09].

Control is like Cadoz's concept of modulation that dynamically modifies the physical phenomena initiated by the excitation component of gesture: it may be continuous or discrete.

Movement type characterizes a gesture by its *external characteristics* in terms of duration and acceleration variation. It may be impulsive of fluid. *Impulsive movements* are characterized by strong variation in a small amount of time whereas *fluid* movements are much slower and characterized by small acceleration variation over a larger amount of time.

In Chapter 3, which proposes a novel taxonomy for accelerometer-based gesture with hand-held devices, I will address these two aspects of dynamics, control and movement type, in a more generic, fine-grained, and uniform manner.

So far, we have analyzed gestures along two complementary perspectives: function (Section 2.1) and morphology or styles (Section 2.2). In the context of computer human-interaction, the functional and morphological properties of gestures, which are produced *with* or *through* input devices, are necessarily correlated with the nature of input devices. This dimension is covered, but overlooked in Karam's *et al.* taxonomy. Baudel [Baudel 95], then Roudaut [Roudaut 10], and more recently Hinckley [Hinckley 11], have considered input technologies in more details.

2.3. Gestures and enabling technologies

2.3.1. Gestures and enabling technologies according to Baudel

In his thesis, Baudel highlights the limitations and advantages of different input devices in relation to gestural interaction techniques [Baudel 95]. Limiting the domain of interest to *haptic* inputs devices, he defines three main groups: (1) **simple haptic** input; (2) **mark and traces** recognition; and (3) **"pure" gestural** interaction.

Simple haptic input devices support interaction driven by discrete states changes produced by a user on input sensors such as keyboards and mice. This is a well-known class of interaction where the semantics of the user's action is described by discrete states of input devices. Reusing Baudel's examples in [Baudel 95], let us consider drawing a rectangle with a mouse. The first anchor point is specified when the user presses the mouse button down while the second anchor point is acquired when the button is released. The trajectory performed between the two anchor points is not relevant for identifying the semantics of the actions: the gesture is represented by those two discrete points denoted through two distinct mouse button states (pressed, released). Because of the dynamics of the interaction has no effect on the semantics, Baudel does not consider this class of haptic interaction as gestural.

Mark and Traces devices come into play when the trajectories of the whole interaction has an impact on the semantics of the interaction. The starting point and the end point play a marginal role, while the trajectory prevails. Consider two marks, the first one shaped as a "C" corresponding to the *draw circle* command while the second is shaped as an "L" corresponding to the *draw rectangle* command. During the interaction, the user can designate both the same starting point and ending point, but the followed trajectory will identify the desired command. This class of gestural interactions is characterized by the importance given to the dynamics. The mouse and the stylus best represent the class of input devices that support marks and traces interaction techniques.

"Pure" gestural devices such as cameras, DataGloves or eye trackers, take into account more than just samples of the dynamics. They are capable of sensing *additional information*, such as which hand is involved in the interaction, thus freeing the user from any intermediate language translation and implementing *direct gestural interaction*.

Baudel is then able to show the correlation between each class of input devices and the expressive power of gestures produced with this class to command a system. The expressive coverage of a gesture may be either one of the following:

Simple designation either of the command or of the parameters: Common WIMP components fall in this category (buttons, scrollbars or menus). They represent a command, and their functional behavior expresses the arguments (when available) by simple designation.

Simple designation of the command & step-by-step parameters selection: More complex WIMP widgets implement this interaction. In drawing editors, for example, the selection of the component to draw (the command) is followed by the set up of the parameters (color, line style, start point, end point...).

Step-by-step command selection & simple parameters designation: The parameters (e.g., icons) are designated before the command to be applied (as a menu item or double click).

Step-by-step command and parameters selection: The specification of the command together with the parameters are bundled together (as for Marking menus).

Handwriting recognition: Letters or symbols are used to encode the parameters that are needed by a global command which, in turn, is in charge of decoding the inputs.

Symbols recognition and match up: This class represents complex gestural interaction systems capable of recognizing articulated gestures, of breaking them down into primitives and interpret them in order to accomplish the desired semantic action (command+parameters).

Compared to early work in psycholinguistics and Karam's *et al.* taxonomy for gestures, Baudel's analysis is clearly driven by technical considerations as illustrated by his concerns for the mapping problem of "device-gesture" with a "complete system command". Roudaut has a similar approach for mobile and hand-held devices [Roudaut 10].

2.3.2. Gestural interaction techniques according to Roudaut and Hinckley

Roudaut re-uses the notion of interaction modality as defined by Nigay to denote the couple "device-gesture". In [Nigay 96], an interaction modality is defined as a couple

	Device	Interaction language
Movements modality	Touch screen Mouvements sensors Keyboard	Deictic gesture Semaphoric gesture Physical gesture
Visual modality	(Touch) Screen Complementary rendering device	Temporal multiplexing Space multiplexing Depth multiplexing

Table 2.1.: Roudaut's Gestural interaction language applied to mobile devices platforms [Roudaut 10].

$< device, language >$ where "language" denotes an interaction language. Sentences that are compliant with this language can be mapped onto system commands. Inspired from Wexelblat and Karam's work, Roudaut defines three types of interaction languages.

Deictic gestures aim to characterize a point in space. Pointing a target on a touch screen is a *deictic gesture*. The gesture can take advantage of more complex interactions to implement either a Drag & Drop interaction technique or a scroll behavior.

Semaphoric gestures use symbols as units of information. They are either cultural symbols (cf. emblems), analogue to letters or numbers, or more abstract ones, like lines or circles.

Physical gestures use real world human articulation capacities in order to propose *natural* interactions. *Embodied user interfaces* [Fishkin 98] lie in the category.

Table 2.1 shows examples of mappings between devices and interaction languages.

2.3.3. Gestures and enabling technologies according to Hinckley

In a recent work, Hinckley *et al.* have related gestures to motion sensing technologies [Hinckley 11]. Informed by the seminal work of Card *et al.* on input devices [Card 91], they propose a two axis taxonomy: the property sensed by the technology and the activation mechanism employed to trigger system interpretation.

The property sensed includes acceleration, angular velocity, vibration, sheer torque, and position-based motion sensing (as provided by a DataGlove or a video camera).

	INDIRECT		DIRECT		
	Mech. Buttons	Indirect Touch & Pressure	Direct Touch	None (pure motion)	
Acceleration (linear accelerometer)	Virtual Shelves [9]		***Pivot-to-lock***	Auto screen rotation [19]	Absolute (Θ)
	Tilt for text [37]	Tilt scrolling [14,19]	***Tilt-to-zoom*** Measuring wrist angles [29]	Rock n Scroll [1]	Relative (ΔΘ)
	Expressive Typing [22]		***Hard-tap*** ***Hard-drag*** Piano forte [12]	Whack gestures [21]	Hard contact force
			Hold+shake ***Tip-to-select*** Chucking [17]	TimeTilt [30]	Motion gesture
		Graspables [36] Grip sensing [24]			Stability (no motion)
Angular Velocity	Deblur image on shutter press (accel. + gyro) [23]		***Thumb vs. finger usage***	***Tapping corners*** Dual-screen Reader [7]	
Vibratory		Scratch input [15]	Skinput [16]	Bone conduction microphones	
Shear, Torque		Gummi bendable computer [34]	Vector touch-screen [18]		
Position-based motion sensing	Chameleon [13] (Hold button + sensed motion of device)		TouchProjector [2], Spilling [28], (Direct input on screen + sensed motion of device)	PhoneTouch [32] (motion signal on phone, at same time as phone contact with surface)	

Figure 2.5.: Hinckley *et al.* propose a complete design space of motion sensing interaction techniques [Hinckley 11]. Highlithed the section of the design space this thesis focuses on.

Clearly, the nature of this dimension is very much influenced by Card's *et al.* model of input devices.

The activation mechanism may be direct (as direct-touch contact with a display), or indirect through mechanical buttons as well as through "touch, pressure, or grip sensors integrated in the device". The absence of activation mechanism corresponds to continuously active motion sensing. It is classified under direct activation mechanism and denoted as "pure motion".

As Figure 2.5 shows, Hinckley's *et al.* classification space goes one step further than Roudaut's mapping by decomposing input devices in terms of the properties sensed. It also shows the focus of my own work on accelerometers and touch screens. On the other hand, the taxonomy does not go into any detail about the type of data sensed by the accelerometers or by the touch screen (which may be multi-touch). I will address these issues in the next chapter.

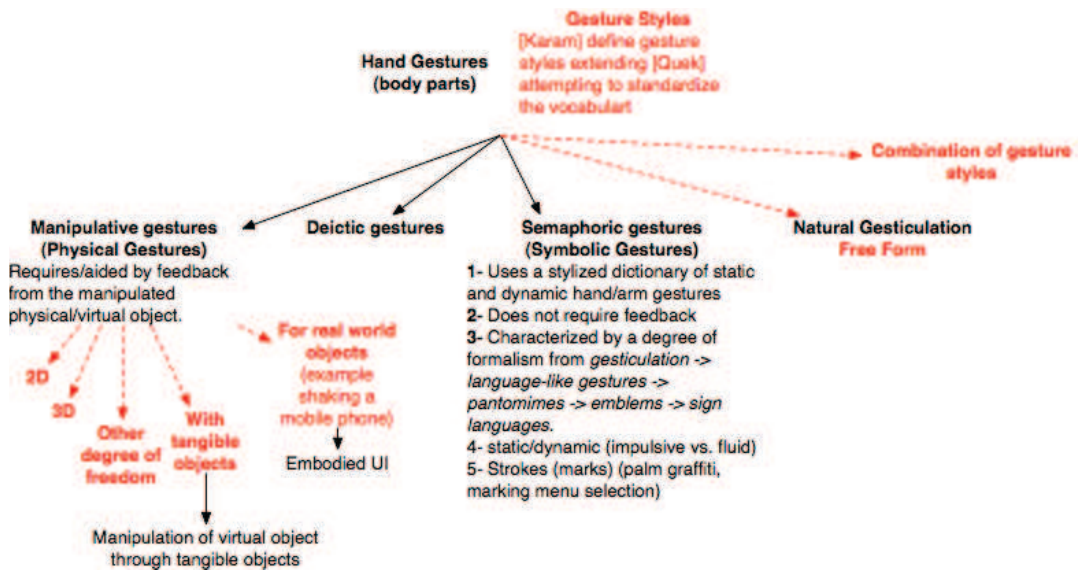


Figure 2.6.: A synthetic representation of the gestures terminology used in different taxonomies.

2.4. The Need for a Unified View

As demonstrated in this chapter, gesture has been studied from different perspectives (function, morphology, and relationship with the technology), and by distinct scientific communities including linguistics, cognitive psychology and computer human interaction. It results from this diversity a terminological jungle that makes it hard for the neophyte to grasp the domain. Typically, different terms are used to denote the same gesture style. For example, symbolic gestures are also called iconic gestures, and manipulative gestures are also called physical or instrumental.

I propose Tables 2.7 and 2.8 as well as Figure 2.6 as representations of a unified and synthesized view of the terminology used for gestures. Figure 2.6 shows a complementary view of the hierarchical relationships between the gesture styles along with the key contributors in the domain. Table 2.7 shows the correlation of the terminology developed in Human-to-Human communication with that of Computer-Human Interaction. Table 2.8 correlates the level of formality of gestures with its expressive power.

The correlation between the terminologies is represented in Table 2.7 by the two main columns entitled “Human-to-Human communication” and “Computer-Human Interaction”. In addition to vocabulary elements, each cell of the table contains a definition illustrated with examples of gestures selected from the state of the art. The two primary rows of Table 2.7, “Conversation metaphor” and “Direct manipulation-model world metaphor”,

correspond to the two major metaphors used in HCI [Hutchins 85]. (A conversation metaphor, as exemplified by the Unix shell, implies the existence of a language which acts as a barrier between the user and the world of interest. With the model world metaphor, as exemplified by the desktop metaphor, the world is represented explicitly and the user can act on it directly, creating the sensation of direct engagement, thus the expression "direct manipulation".) A coarse grain analysis of the two main rows shows that the vocabulary is richer for the conversation metaphor than for the direct manipulation-model world metaphor. This lack of balance indicates the need for research in the area of gestures for manipulative/physical/instrumental interaction.

The primary distinction between "manipulative/physical, instrumental gestures" and gestures used in a conversation metaphor is the necessity for a tightly-coupled feedback for control (cf. second column from the left side of the table): while manipulating an object, whether it be physical or virtual, the user needs to perceive an immediate and continuous system reaction in order to modulate the gesture. Within the conversation metaphor, gestures are distributed across three classes depending on their degree of formality: they may be codified, partly codified, or not codified. Whereas Human-to-Human Communication introduces a rich vocabulary for each one of these categories, the Human-Computer Interaction side ignores these subtleties. Typically, the HCI semaphoric gestures abstract away the set of pictographs, kinetographs, ideographs/metaphorics, and emblems identified in Human-to-Human communication. Whereas McNeill (who belongs to the Human-to-Human side) makes a distinction between gesticulation and pantomime (both of them are global gestures but gesticulation is synthetic - top down, while pantomime is analytic - bottom up), Karam (on the Human-Computer side) bundles them together as one single element: gesticulation.

By contrast, the Human-to-Human communication side, which has studied gesture in combination with speech, does not provide any term for the various forms of direct manipulation. The concept of object-adaptor introduced by Ekman and Friesen (see Section 2.1) to denote the manipulation of a physical object, is not intended as a communicative means but is related to stress and cognitive engagement.

Interestingly, there is a clear consensus for the deictic gesture which sits between the conversation metaphor and the direct manipulation gestures with an exception for Baglioni who considers deictics as part of physical gestures.

The bottom row of Table 2.7 reflects the combination of gesture styles. The Human-to-Human side is concerned with gestures that accompany speech, whereas the Human-

				Human to Human communication			Computer-Human Interaction				
									Hand-held devices		
				Efron	Ekman et al.	McNeill et al.	Quek et al.	Karam et al.	Roudaut et al.	Baglioni et al.	Cadoz
Conversation metaphor (faire-faire)	In general, no need for tightly coupled feedback	Codified	Portray concrete idea	Physiographic		Iconics	Semaphoric	Dynamic semaphoric plus strokes and traces a) rectilinear marks b) free-form paths	Semaphoric	Symbolic	
				a) iconographic *Physical shape imitation, e.g., Drawing shape of an object or spatial relationships*	Pictographs						
				b) Kinetographic *Bodily action imitation, e.g. "I write to him"*	Kinetographs						
			Portray abstract idea	Ideographic *Traces or sketches out in the air the paths of direction of the thought pattern*	Ideographs	Metaphorics				Metaphoric *Indirect analogy with the real world, e.g., TiltText*	
		Partly codified	Segmented synthetic gesture	Emblematic *e.g., the OK sign*	Emblems	Emblems		Static Semaphoric		Abstract *Arbitrary mapping with meaning, i.e. all gestures that are not symbolic, metaphoric, physical, e.g., Connect*	
		Not codified	Global synthetic gesture			Gesticulation		Gesticulation			
			Global analytic gesture			Pantomine					
			Pointing	Deictic	Deictic	Deictic	Deictic	Deictic	Deictic	Deictic	
Direct manipulation, model world metaphor (faire)	Need for tightly coupled feedback						Manipulative	Manipulative *(with refinements in 2D/3D, with physical objects, etc.)*	Physical	Physical *Direct analogy with the real world gestures and their effects, thus includes deictic, e.g. MotionLens, WiiTennis*	Instrumental *combination of epistemic, ergotic, semiotic functions*
											Excitation gesture
										control *continuous VS discrete*	Modulation gesture
											Selection gesture
				Combination of gesture with speech			*Combination of gesture styles, and multimodal interaction*				

Figure 2.7.: An integrated view of the analyzed taxonomies highlights the relationshinps among different approaches and metaphors.

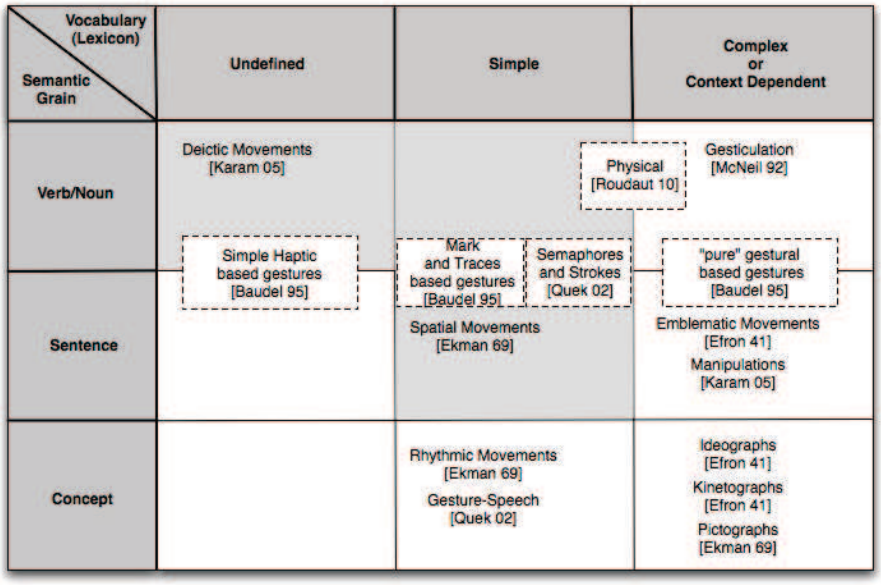


Figure 2.8.: Correlation between gesture formality and expressive power.

Computer side is opened to any type of combination between gestures styles as well as with any other modalities. I will address multimodal interaction involving gesture in Chapter 7.

Unless otherwise specified, I will subsequently refer to Karam's et al. terminology to qualify gesture styles. Using Karam as a reference, then my research is primarily concerned with deictic and semaphoric gestures, but it involves some aspects of manipulative gestures as well. Therefore, I am not concerned with gesticulation although gesticulation, which is not codified, is supposed to represent the ultimate natural form of gestural interaction. I will show in Chapter 6 and Chapter 7 how the combination of visual feedback and feedforward supports the illusion of freedom.

Figure 2.8 shows the relation between gesture formality (from non-codified free-form to fully-codified by a grammar) and the semantic expressive power of gestures. With regard to expressive power, a single gestural act may be able to specify one item only (e.g., an object of interest or a command name), or it may convey a command along with its parameters, or it may express a complex thought as a set of several sentences. For example, a deictic gesture is not codified but can specify only one item at once. *The grey area of Figure 2.8 shows the coverage of this doctoral research: from "non codified and 1-item-specification" to "partly codified covering 1 command-and-its-parameters-specification".*

2.5. Synthesis

This chapter has shown the breadth of the research related to gestures by providing an overview of the key terminology-oriented taxonomies, ranging from psycholinguistics to computer science. I have proposed a synthetic view of the terminologies by correlating the multiple perspectives developed in the state of the art. Within this big picture, my doctoral research *addresses the particular case of deictic and semaphoric gestures for computer human interaction whose expressive power may range from the specification of a single token to the expression of a command along with its parameters. Because I address the particular problem of gesturing with accelerometers-enabled handheld devices, I am also concerned with instrumental interaction.*

The pioneering work on accelerometer-based interaction techniques [Fitzmaurice 93, Hinckley 00, Levin 99, Partridge 02, Rekimoto 96] has paved the way for an active research area [Ballagas 06, Williamson 07, Wilson 03]. Although these results satisfy "the gold standard of science" [Shaw 03], in practice, they are too "narrow truths" [Brooks Jr 88] to support designers decisions and researchers analysis. Designers and researchers need an overall systematic structure that helps them to reason, compare, elicit (and create!) the appropriate techniques for the problem at hand. Taxonomies, which provide such a structure, are good candidates for generalization in an emerging field. The taxonomies I have presented in this chapter are appropriate for comprehending the breadth of the domain, but not for reasoning at a fine grain about the design of a gesture-based interaction technique. In the next chapter, I propose a novel taxonomy for gestural interaction techniques based on accelerometers.

Chapter 3.

A taxonomy for gestural Interaction Techniques based on accelerometers

I propose a new taxonomy for gestural interaction techniques based on accelerometers. The motivation for limiting the coverage of the taxonomy to accelerometers-based interactions is that gestural interaction for mobile devices is a very vivid and unstructured area of research. In addition, accelerometers are currently the most pervasive technology for sensing multiple dimensions of actions in the real-world [Hinckley 00]. The challenge is to provide a classification framework that is both complete and simple to use. Since completeness is illusory in a moving and prolific domain such as user interface design, I have not included it as a goal. I show, however, that the taxonomy is able to go beyond accelerometers-based techniques, covering a wide domain of issues related to Human-Computer Interaction.

To develop this taxonomy, I have built a controlled vocabulary (i.e. primitives) obtained through an extensive analysis of the taxonomies that have laid the foundations for HCI more than twenty five years ago. For the most part, this early work in HCI has been ignored or forgotten by researchers driven by the trendy "technology push" approach.

My taxonomy is based on the following principles:

1. Interaction between a computer system and a human being is conveyed through input (output) expressions that are produced with input (output) devices, and that are compliant with an input (output) interaction language.

2. As any language, an input (output) interaction language can be defined formally in terms of semantics, syntax, and lexical units.

3. The generation of an input (output) expression involves using devices whose characteristics, from the human perspective, have a strong impact on the expressiveness and the effectiveness of the user interface [Buxton 83].

Building on Foley's work [Foley 90b] as well as on Buxton's pragmatics considerations of input structures [Buxton 83], my taxonomy brings together the four aspects of interaction ranging from semantics to pragmatics with the appropriate human-motivated extensions for addressing the specificity of gestural interaction based on accelerometers. In contrast to Mackinlay's *et al.* semantic analysis of the design space for input devices [Mackinlay 90], I do not consider the transformation functions that characterize the system-oriented perspective of interaction techniques.

This chapter is organized as follows: First, I review the taxonomies that have served as sources of inspiration for my own work: Foley's taxonomy for having identified the generic basic tasks supported through graphical user interfaces, and Buxton's and Card's *et al.* taxonomies for their device oriented concerns. The analysis of these taxonomies offers the opportunity to clarify the terminology (after all, what is an interaction technique?) Then, I apply the proposed classification space to well-known mouse-driven interaction techniques. In the following chapter, I present my taxonomy illustrated with a survey of accelerometers-driven gestural interaction techniques. I conclude with future directions for research that my taxonomy has permitted to discover. The expectation is to provide new insights and to start promising directions for the design of novel and powerful gestural interaction techniques.

3.1. The Foundations

Classic HCI papers propose a wide spectrum of taxonomies falling into one of two main categories: **linguistics**-inspired taxonomies, and **morphological** taxonomies. Linguistics inspired taxonomies are driven by the lexical, syntactic, and semantic structures of languages. In the morphological approach, interaction techniques are points in a multi-dimensional space where each dimension represents a differentiating property. The next paragraph analyzes both approaches by referring to existing examples.

Tasks	Requirements
Select	Size of set, if fixed Range of set, if variable
Position	Dimensionality: 1-D, 2-D, 3-D Open loop or Closed loop. Resolution
Orient	Degrees of freedom: 1, 2, 3 Open loop or Closed loop Resolution
Path	Maximum number or path elements to be retained Type of interval between each element on path Size of interval between each element on path Dimensionality: 2-D or 3-D Open loop or closed loop Resolution Type: position or orientation or both.
Quantify	Resolution Open loop or Closed loop.
Text	Size of character set Maximum length of string.

Table 3.1.: Foley's classification of fundamental interaction tasks expresses the requirements that interaction techniques must satisfy [Foley 90b].

3.1.1. Linguistics-inspired taxonomies

Historically, the semantic-syntactic-lexical layers developed for artificial (formal) languages, have served as a useful tool for structuring the design process of user interfaces. In this approach, a user interface is assimilated to an artificial interaction language. This language is composed of an input interaction language that allows users to express their mental goals, and of an output interaction language that expresses the system state in terms that match the user's conceptual model. Moran's *et al.* Command Language Grammar (CLG) [Moran 81] Foley, as well as Wallace and Chan's taxonomy [Foley 90b] are examples of this approach.

The main contribution of Foley *et al.*'s taxonomy is two-fold:

(1) **Six interaction tasks** that define the semantics of a canonical set of non-terminal symbols [words] for graphics (*select* "to make a selection from a set of alternatives", *position* to "indicate a position on an interactive surface", *orient* to "orient an entity in 2-D or 3-D space", *path* to "generate a path, which is a series of positions or orientations created over time", *quantify* to "specify a value, or quantify a measure", *text* to "enter a text string").

(2) **The cross product of these six interaction tasks with input devices** that shows the many ways each interaction task can be performed with existing devices.

To complement the six interaction tasks, which are tasks for specifying something, Foley introduces **four controlling tasks** such as *Stretch* and *Sketch* to express direct modifications of entities. Overall, Foley *et al.*'s taxonomy describes a large number of interaction techniques but does not define sharp boundaries between input devices, interaction tasks and controlling tasks resulting in an ambiguous definition of the notion of interaction technique *per se*.

For this reason, I define the terminology used in the linguistics approach to user interface design in the following way:

- a complete [input] sentence such as "<move> <entity> <position>" instructs the computer system to perform a function (ideally, this function implements the semantics of the sentence).
- a sentence is composed of words (e.g., <move>, <entity>, <position>) whose assembly is compliant with a predefined syntax where each word is a symbol, that is, a primitive non-terminal that conveys a unit of semantics (or lexeme).

Using these definitions, "the entry of each symbol [word] by the user is an *interaction task* performed by means of an interaction technique" [Foley 90b]. In other words, an *interaction technique* produces *non-terminal symbols [words]* by assembling terminals according to predefined lexical rules. These *terminals*, which belong to the digital world, result from the transformation of physical real-world properties and actions sensed by physical input devices.

My taxonomy re-uses Foley et al.*'s interaction tasks as a basis for non-terminal symbols: they are simple semantic units that have proven to be empirically valid.* In particular, Ballagas *et al.* use Foley's *et al.* interaction tasks as a structuring framework to analyze smartphones viewed as input devices [Ballagas 06]. Although these semantic units are empirically sound, their lexical level "lumps together issues as diverse as: how

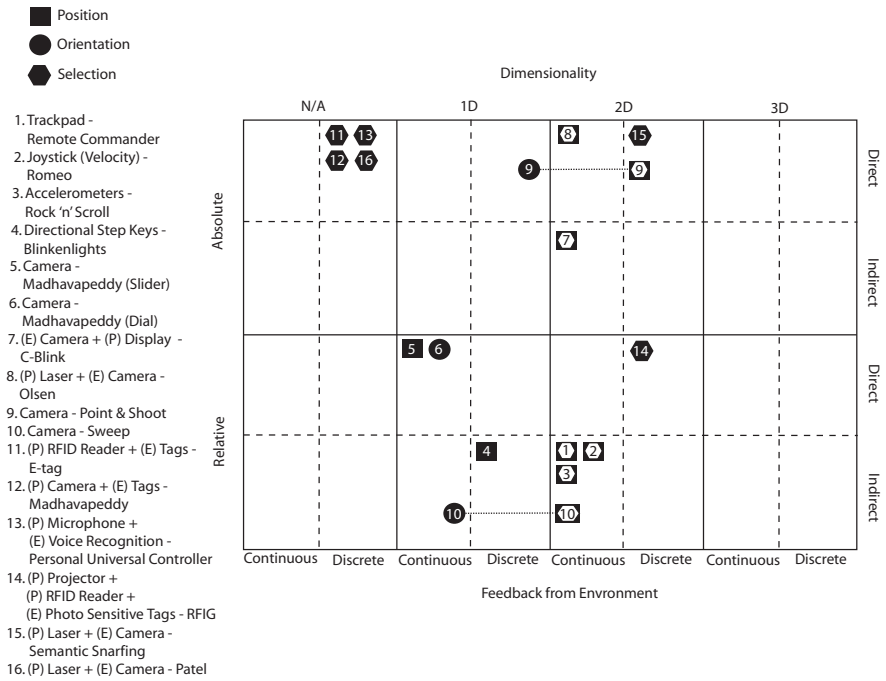


Figure 3.1.: Ballagas taxonomy plots existing interaction techniques implemented by using the phone as input device [Ballagas 06].

tokens [words] are spelt, where devices are placed in the work station, the type of physical gesture used to articulate a token." [Buxton 83].

Consequently, Buxton proposes to make a clear distinction between lexical issues as defined in artificial languages theory (e.g., spelling of words and choice of terminals) from "pragmatic issues of gesture, space and devices" which define "the primary level of contact of a user with a system" [Buxton 83]. Drawing on the importance of pragmatics on users' experience with systems, Buxton proposes a taxonomy of input devices that makes explicit pragmatic attributes including physical properties sensed by input devices (such as pressure, motion and position), the number of dimensions sensed (i.e. the number of degrees of freedom), as well as the sensing type (devices that work by touch vs. devices that require a mechanical intermediary). Buxton's taxonomy helps in finding equivalences between input devices, or in identifying places for the development of new devices. *Drawing on Buxton's analysis, my taxonomy, which is motivated by gestural interaction, incorporates some aspects of pragmatics.*

In the same vein as Buxton, Mackinlay *et al.* [Mackinlay 90], then Card *et al.* [Card 91], extend Buxton's work using a morphological approach to the analysis of input devices [Buxton 83].

3.1.2. Morphological Taxonomies

Figure 3.2 shows Mackinlay's *et al.* taxonomy considered by the scientific community as the archetypal morphological approach to device modeling. Mackinlay's *et al.* model (improved later on by Card *et al.* [Card 91]) uses a primitive movement vocabulary and a set of three composition operators (merge, layout, and connect) which, applied to the primitives, produce a design space for reasoning about input devices. Formally, an input device is defined as a six-tuple composed by <the manipulation operator M, the input domain set In, the current state S, the resolution function R, the output domain set Out, behavior W>. Here detailed:

$$< M, In, S, R, Out, W > \tag{3.1}$$

where:

M, the manipulation operator , represents the physical properties (force or position) sensed by the input device, and whether these properties are linear or rotary, absolute or relative,

In, the input domain set of the physical properties sensed by the input device,

S, the current state of the device,

R, the resolution function maps the properties from the input domain set into the output domain set,

Out, the output domain set denotes the co-domain of the resolution function,

W, works covers any additional aspects useful for describing the behavior of the input device.

Composition operators are used to combine inputs, outputs and devices, for example,*merge* such that the resulting input domain is the cross product of the input domain of the two devices, *connection* to combine two devices by cascading the output of one device to the input of the other, and *layout* to express spatial relationships between devices.

The resulting taxonomy (Figure 3.2) is a multidimensional parametric space where the y-axis denotes the physical property that can be manipulated by the device (In, M), while the x-axis corresponds to the dimensions of interest during the manipulation (Out, S). Interestingly, the taxonomy gives an idea of the continuity of the interaction

technique supported by the device and the grain of the interaction itself by integrating information about the resolution function (R) that maps the input domain set into the output domain set.

In [Card 91], Card *et al.* present their exploitation of the taxonomy to reason about the effectiveness of input devices in terms of *Desk footprint*, *Pointing Speed*, *Pointing Precision*, *Errors*, *Time to Learn*, etc. These criteria form a sub-framework usable to compare apparently similar input devices. As important, Card *et al.* show how to reason about mappings between input devices and interaction tasks (e.g., pointing task, viewing task). For doing so, interaction tasks as well as input devices are plotted in the design space, while task-device mappings are represented as a connect operator from the device to the task. This representation completed with the parameters of the design space provides the designer with a sound and systematic apparatus for reasoning about the various design options. In other words, a morphological design space like this one can be used to integrate the results from several disciplines. Not only it supports reasoning on existing solutions, but also its layout structure *per se* is intended to foster the discovery of novel solutions.

More recently, Nancel *et al.* [Nancel 09] proposed a morphological taxonomy for reasoning about menu-based techniques applicable to pen-driven interaction. This approach has been extended by considering additional input sensors. This work focuses on the property sensed by the considered devices and uses the vocabulary introduced by Mackinlay [Mackinlay 90]. The originality of this taxonomy comes from the idea that by choosing the input device, the design space becomes a classification of input techniques. On the other hand, the proposed organization limits the discussion to the lexical aspect of interaction, leaving aside the syntactic and semantic dimensions. This limitation is too restrictive when considering gestural interaction techniques. In particular, context, which is key in mobile computing [Coutaz 05], as well as the distinction between foreground and background interaction [Buxton 95] are ignored.

As discussed in the previous chapter, Hinckley *et al.* have proposed a design space inspired from Card's *et al.* morphological approach, that portrays the various forms of synergistic use of motion and touch [Hinckley 11]. Although the space covers different input technologies to sense motion, it does not cover the pragmatic details of interaction nor the notion of context.

As shown in Figure 3.3, the same gesture may convey very different meanings depending on the context in which it is produced: "go to previous photo" as for the

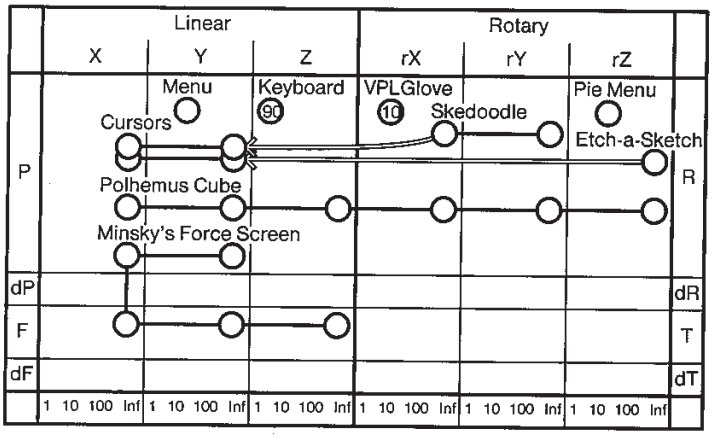


Figure 3.2.: Physical, virtual and composite input devices classified within Mackinlay's *et al.* taxonomy. A circle in a cell indicates that a device senses a physical property characterized by the coordinates of the grid. A black line represents a merge composition. An arrow represents a connect composition. A dashed line - no example shown here - a layout composition [Mackinlay 90].

Apple's photo album (or "go to next slide" as in Charade in [Baudel 93]), "open a submenu" in Francone's Wavelet Menu [Francone 09], or "unlock" the iPhone screen. In addition, a gesture that makes sense for the system, may not be acceptable in a public social context [Rico 10] as it could be meaningful and interpreted by the public itself.

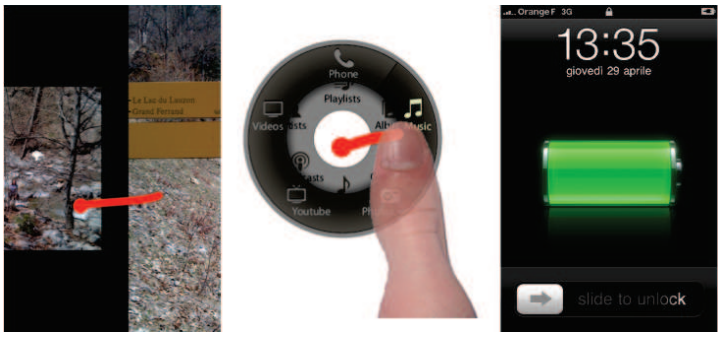


Figure 3.3.: The "sliding" gesture is semantically multiplexed to achieve different meanings, depending on context.

These observations motivate the definition of a new taxonomy presented next.

3.2. A New Taxonomy for Accelerometer-based Gestural Interaction

My taxonomy captures the following requirements:

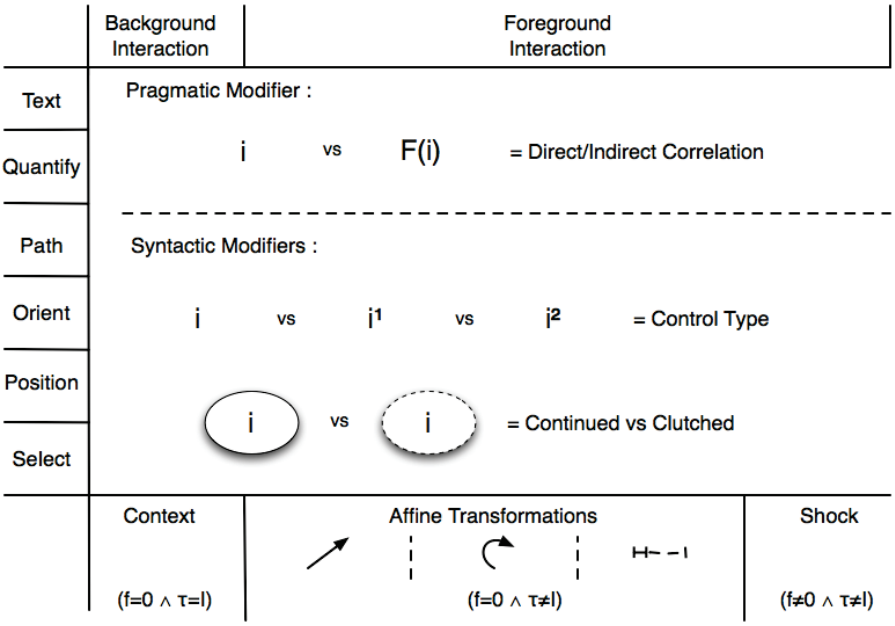


Figure 3.4.: A new classification space for gestural interaction techniques based on accelerometers. The abscissa defines the lexicon in terms of the physical manipulations users perform with the device, with a clear separation between background and foreground interaction. The ordinate corresponds to Foley's interaction tasks. An interaction technique is uniquely identified by an integer i and plotted as a point in this space. Each point is decorated with the pragmatic and syntactic properties of the corresponding interaction technique. There are two syntactic modifiers: an oval indicates whether the interaction technique is clutched or continued, and an exponent expresses the control type (position, speed, or acceleration). F, which is the only pragmatic modifier, indicates the degree of indirection of the interaction technique.

1. Coverage of semantic, syntactic, lexical, and pragmatic issues of interaction where semantic granularity is that of Foley's et al. interaction tasks;

2. Adoption of a user-centered perspective where physical human actions are premium, leaving aside the internal computational transformations;

3. Consideration for context; Coverage of both foreground and background interaction (as defined by Buxton [Buxton 95]).

Figure 3.4 shows the elements of the framework which are described next. (Figure 3.5 illustrates the use of the taxonomy for conventional WIMP techniques, whereas novel accelerometers hand-based interaction techniques are presented in Figure 4.3 in the next chapter.)

3.2.1. Lexical Axis

Because of the focus on users' involvement in the interaction, the input lexicon corresponds to the physical actions users apply to devices. Human physical actions are organized in two distinct groups: (1) conscious actions that belong to the foreground interaction, and (2) unconscious actions that correspond to background interaction. The foreground interaction area contains the interaction techniques that require the user to consciously manipulate the device to reach some objective (as for the sliding gesture of Figure 3.3). The background interaction area corresponds to the interaction techniques where the system interprets user's unconscious actions together with contextual information to perform some system state change on behalf of the user. For example, during a phone call, the iPhone switches off the screen backlight to save battery life as the user brings the device next to the ear.

Whether human actions are performed consciously to address the system or not, the classification space characterizes these actions with two additional variables: (τ) the geometrical transformation matrix that models user's movements in space, and (f) the frequency of these movements. The combinations of τ and f identify three sub-areas within the lexical axis: "Context", "Affine Transformations" and "Shock". The affine transformations group identifies the most common interaction techniques based on translations, rotations and/or scales (in this case, τ is different from the identity matrix I), and without any repetition (that is, f is equal to zero, meaning that the interaction is time driven). The sliding gesture of Figure 3.3 falls in this category. The shock category identifies those interaction techniques based on a combination of translations, rotations and/or scales (τ is different from the identity matrix) repeated over time (then, f is different from zero). The shake gesture exemplified by Shoogle [Williamson 07] falls in this category. The context category corresponds to unconscious human manipulations that the system may interpret to feed into its own context model and, depending on this context, acts on behalf of the user. For this situation, I stipulate that τ is the Identity matrix and f is equal to zero.

As a simple example, consider the physical actions users need to perform in order to close a graphical window in a conventional WIMP environment. First, they have to move the mouse in the physical world, then to press the mouse button when the final position is reached in order to trigger the close window command. Both of these actions involve a translation: on the mouse plan for the first action and on the perpendicular

axis when pressing the mouse button down. Yet both of them are time driven and do not involve frequency.

3.2.2. Syntactic Axis

Independently from the device used, the syntactic dimension of an interaction technique is characterized with the following two variables that I call *syntactic modifiers*: (1) the existence (or absence) of triggers (or clutch) to specify the begin/end of the interaction, and (2) the control type associated with the input token, which may be position-control, speed-control or acceleration-control.

Analyzing the syntactic dimension of the close window example, mouse movements are translated into pointer movements. This pointer is position-controlled and the interaction is unclutched i.e. there is no explicit start/end action to bind the human physical movements with the software cursor movements. The binding is always on. On the other hand, the selection of the close window button widget, which requires a shock action on the physical button of the mouse, is a clutched interaction technique.

Note that Cadoz's notion of "modulation gesture" introduced as one component of an instrumental gesture (cf. Chapter 2) is now formally described by the "control type" of my taxonomy. In addition, the taxonomy reflects the "activation mechanism" used by Hinckley's *et al.* taxonomy as the existence (or absence) of a clutch, elements that are not expressed explicitly in Roudaut's and Baglioni's taxonomies.

As a result, given that, in the taxonomy, an interaction technique is uniquely identified by an index i, the trigger syntactic modifier is represented as an oval that surrounds the interaction technique identifier using a dashed-line or a continuous line to respectively denote the presence (i.e. clutch) or absence (i.e unclutch) of a trigger. In addition, a derivative-like notation is used to convey the control type where i is decorated with an exponential number that expresses the derivative order with respect to time (i.e. no derivative for position, first order derivative for speed, and second order derivative for acceleration).

3.2.3. Semantic Axis

As justified in the review of the foundational taxonomies developed in HCI, I re-use Foley's interaction tasks: Select, Position, Orient, Path, Quantify, and Text [Foley 90b] (See the vertical axis of Figure 3.4).

Analyzing the semantic dimension of the close window example, the translation of the mouse corresponds to the user's goal to assign a new position to the mouse pointer motivated by the need to select the "close window" button widget. Finally, the click of the mouse physical button corresponds to the goal of confirming the selection of the widget soft button.

3.2.4. Pragmatic Axis

One of the originalities of the taxonomy is the attempt to classify gestural interaction techniques in close connection with their meaning in the user's real world. To do this, I introduce a *pragmatic modifier* that expresses the directness [Norman 86a, Beaudouin-Lafon 00] of the mapping between the user's expectation (i.e. goal) and the semantics of the interaction technique in the computer world. For indirect mapping, the identifier i of the interaction technique becomes the parameter of a function $F(i)$ to indicate the existence of one or several reinterpretation layers, whereas for direct mapping, i does not receive any additional decoration.

Analyzing the pragmatic dimension of the close window example, "positioning the pointer" has a direct pragmatic connection: every physical mouse translation is associated to a pointer translation. "Associating the mouse button to the software button widget" is also characterized by a direct pragmatic connection. The whole sequence of actions on the other hand is characterized by one degree of indirection since the user's objective is to close a graphical window, not to click a software button widget. Therefore, the button widget creates one level of indirection between human actions and the meaning of the sentence.

A number of familiar mouse-driven interaction techniques are now discussed to illustrate the coverage and use of my taxonomy. Gestural accelerometer-based interaction techniques will be considered in the following chapter.

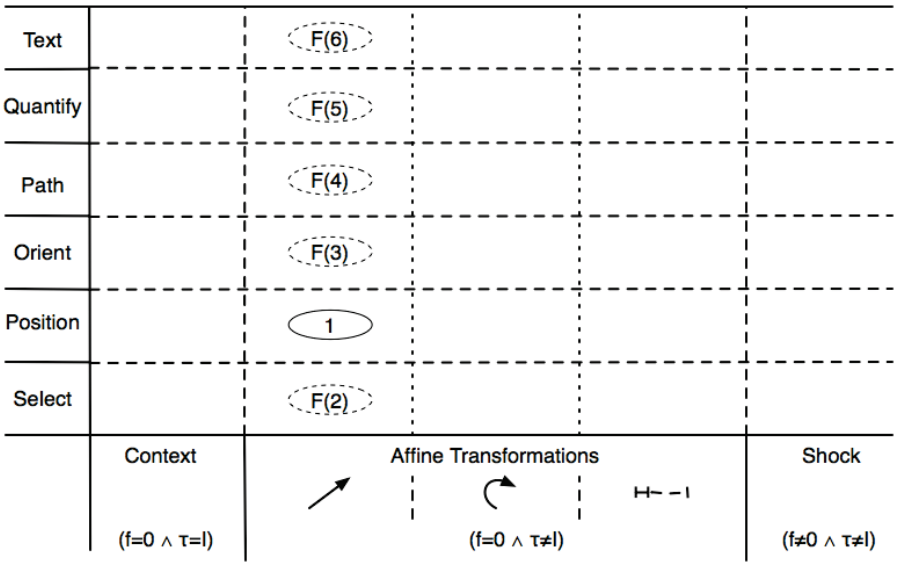


Figure 3.5.: Classical mouse-driven interaction techniques within my taxonomy: (1) Positioning the cursor; (2) Menu item selection; (3) Defining orientation in a graphics editor; (4) Sketching using a drawing tool; (5) Defining a quantity through a slider; (6) Typing text with a virtual keyboard.

3.3. Classification of WIMP Techniques

For conventional GUIs, cursor control is the main mouse-based interaction technique. As discussed above, when specifying a position with a mouse, physical translations are mapped directly into translations of the pointer: there is no pragmatic indirection function F. The interaction technique is position-controlled: there is no syntactic derivative modifier. In addition, it is continuously active. Figure 3.5 shows its corresponding location as interaction technique (1).

Other familiar GUI interaction techniques include selecting an item in a linear menu, changing the orientation of a graphical object within a graphics editor, quantifying a dimension through a slider, or typing text with a virtual keyboard - respectively denoted in Figure 3.5 as interaction techniques (2), (3), (5), and (6). All of them are characterized by some level of indirection due to the use of an intermediary graphical widget. All of them are position-controlled since they are built on the elementary positioning interaction technique. All of them are clutched since they need a trigger to specify the beginning and the end of the interaction (with a the mouse click button). As for Foley's path task *digitizing a sketch* (cf. (4) in Figure 3.5), it consists of a temporal series of positions and orientations. It is thus position-controlled, clutched by the mouse button, and indirect.

The classification of the common mouse-driven interaction techniques within my taxonomy calls for the following observations: (1) the mouse supports all of the interaction tasks identified by Foley, whether it be directly or indirectly. In other words, the mouse can be used to fully control a WIMP-based graphical system. (2) The taxonomy demonstrates the simplicity, the uniformity and the completeness of the interaction language supported by the mouse-driven WIMP interaction techniques since all of them are characterized by the same syntactic and pragmatic modifiers. (3) The taxonomy highlights the limited initiative the system has in such interactions as they are always explicitly performed by users (Foreground Interaction). (4) From the user's perspective, the interaction techniques that are characterized by some degree of indirection are more complex, while those that are pragmatically direct are simpler. This last observation brings forward a fundamental property of my taxonomy: *the less modifiers an interaction technique is characterized by, the simpler it is from the user's perspective*;

3.4. Synthesis

This chapter has presented a new framework for classifying accelerometer-based interaction techniques. This framework, which is motivated by the foundational contributions in HCI, brings together Foley's generic tasks with the formal lexical, syntactic, semantic and pragmatic dimensions of languages to characterize the physical actions involved in gestural interaction.

System-oriented issues have not been addressed, as I did not want at this point, to differentiate interaction techniques by their implementation characteristics. Granularity, resolution function as well as state machines, have already been taken into account by others [Card 91, Mackinlay 90]. My goal is to complement these taxonomies rather than acting as a substitute.

The taxonomy is radically centered on human physical actions. My research hypothesis is that the physical action is the appropriate atomic level from which novel interaction techniques can be designed to provide system-wide consistent languages with specific attention for gestures involving scale as well as for gestures to specify such as Path, Quantity, and Text.

The classical interaction techniques for desktop devices have been used as a primary use case to demonstrate the capacity of my taxonomy to provide designers with a

synthetic view of the GUI paradigm. In the next chapter, we switch to mobile systems and post-WIMP interaction techniques that use accelerometers as input technology, and see what lessons can be drawn about them from the analysis of the taxonomy.

Chapter 4.

Gestural Accelerometers-Driven techniques: a State of the Art

The accelerometers-based input interaction techniques considered in this chapter are presented in chronological order and plotted in Figure 4.3. For the sake of completeness, all of the variations of an interaction technique are discussed. For example, an interaction technique that exists as continuous (i.e. un-clutched) and clutched appears twice in the taxonomic space, each one denoted with the appropriate syntactic modifiers.

This chapter is structured as follows: first, I apply my taxonomy to a number of representative interaction techniques based on accelerometers. Then, I analyze the picture provided by the taxonomy to define the general frame of the existing interactions. The discussion will drive the reader through the analysis of the state of the art analyzing different approaches to the domain (user *vs.* developer *vs.* researcher). The proposed taxonomy will permit to define the terminology that I will use through this doctoral research.

4.1. Application of the proposed taxonomy

In the last chapter, I presented a new classification space for gestural interaction. In the current section I review and organize the state of the art of the past thirty years of accelerometers-driven gestural interactions.

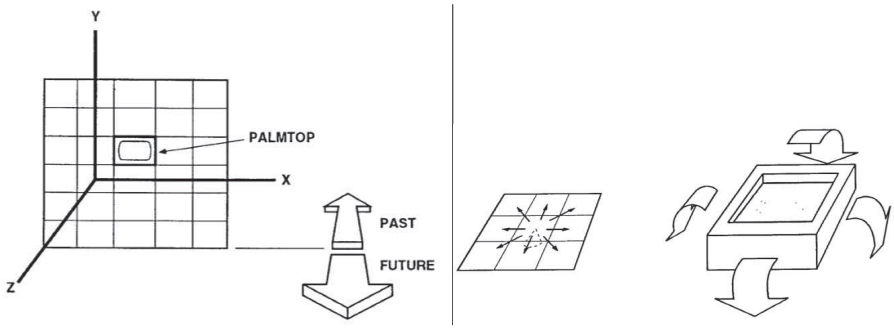


Figure 4.1.: Fitzmaurice et al. used the *Move* action to command the history of their spread-sheet, while the *Tilt* of the device let the user span the cells around the selected [Fitzmaurice 93].

Chameleon

Chameleon is one of the pioneering examples of immersive interaction techniques based on accelerometers *et al.* [Fitzmaurice 93]. It is a palmtop solution where gestural interaction techniques are aware of the spatial position and orientation of the device. Three basic interaction techniques are proposed (denoted respectively as (7), (8), (9) in Figure 4.3): (7) is based on the translation of the device whereas (8) and (9) involve tilting. All of them are available either as continuous or clutched where the begin/end of the interaction is specified through the press/release of a physical button. Consequently, they are represented twice in Figure 4.3 respectively with the continuous line oval syntactic modifier as well as with the dashed-line oval. In the context of a spread-sheet application, selecting a cell with interaction technique (7) is performed by translating the device in the (x,y) plane. A series of translations along the z axis permits to select successive undo and redo commands. Tilting the device in a direction (say, left) permits to preview the (left-)adjacent cell (see (8) in Figure 4.3). Selection and Preview do not imply any level of indirection as the control is directly connected to the item of interest. Interaction technique (9) supports the manipulation of a circular contextual menu to control a text browser or a movie player, thus introducing one level of indirection.

Rekimoto's experiments

Rekimoto analyzes clutched tilting interaction to control linear and circular menus (denoted as (10) in Figure 4.3) as well as a map application according to interaction technique (11) [Rekimoto 96]. Interaction technique (10) associates a tilt angle to each menu item to select a command through the use of the menu widget leading to an

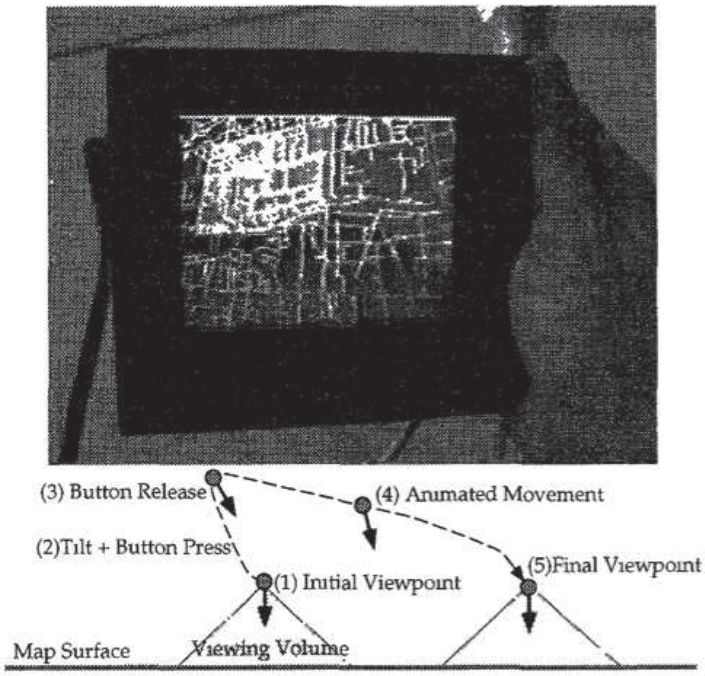


Figure 4.2.: Rekimoto's tilt based menu control the navigation in a map [Rekimoto 96].

indirection. (11) directly associates a physical position to specify a position in a map. There is no indirection. Both techniques are position-controlled and clutched using a press/release of a physical button to mark the begin/end of the interaction.

Harrison's scenarios

Harrison *et al.* [Harrison 98] address the problem of navigation tasks within calendars and text-based applications. The originality of the solution (denoted as (12) in Figure 4.3) relies on the use of a speed control type that changes the syntax of the technique: the larger the tilt angle is, the faster pages are scrolled. It is a clutched interaction technique since the begin/end of the interaction is marked either by repositioning the device to the initial position or by squeezing it. The *squeeze* solution, denoted as (13), is particularly original: it is a meta-interaction technique intended to stop the page-selection interaction technique by shocking the device.

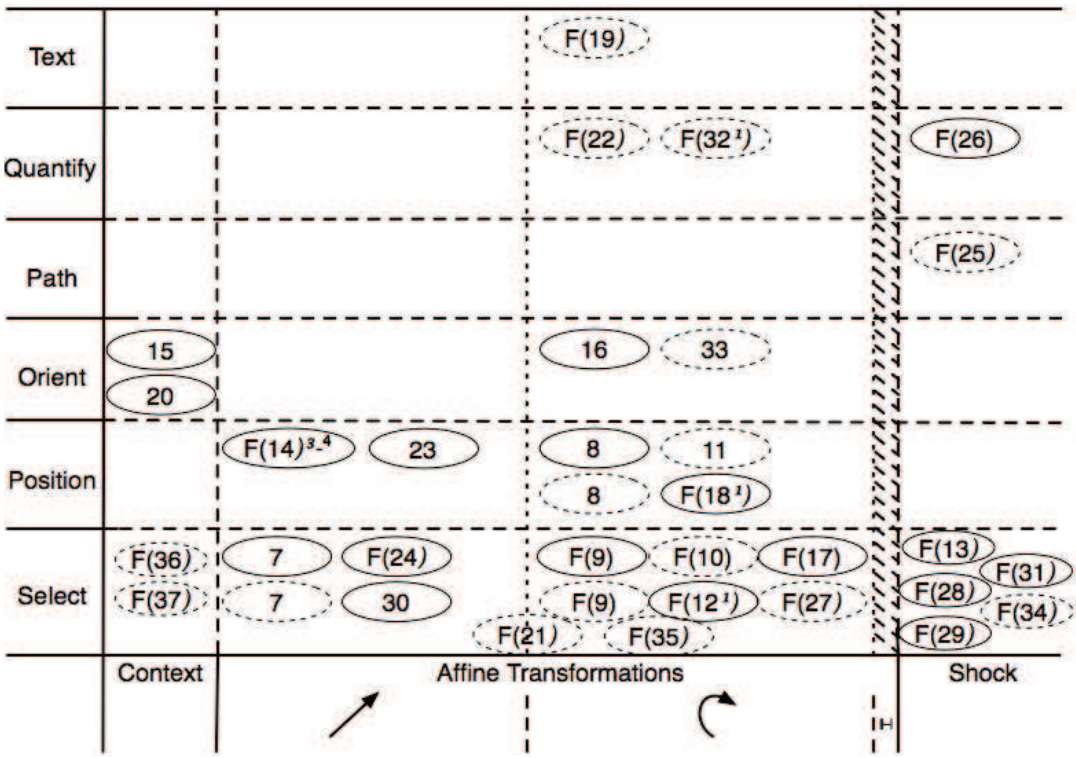


Figure 4.3.: A state of the art of accelerometers-based interaction techniques. An interaction technique is identified by an integer i: (7) successive undo/redo as well as active cell selection through translations; (8) tilt to preview adjacent cells; (9) tilt to select a command in a pie menu; (10) tilt to select commands in linear and pie menus; (11) tilt to control position on a map; (12) tilt to browse a calendar; (13) squeeze to stop an interaction; (14) drawing through physical translations; (15) passive screen orientation adaptation; (16) active screen orientation control; (17) tilt to select pictures; (18) tilt to control first person shooter game; (19) tilt to enter text; (20) passive control of screen orientation and power energy saving; (21) tilt and translation to select physical world object; (22) control volume through tilt; (23) translation of virtual workspace through physical translation; (24) selection of the level of user interface details through translation; (25) gestural authentication with shock durations over time; (26) shake to quantify device status; (27) tilt to select graphical views; (28) shock to trigger an interaction; (29) shock to select the previously active application; (30) gesture recognition; (31) shake to select the next song; (32) tilt to quantify the zoom factor; (33) tilt to control screen/contextual objects rotation; (34) shock the device to select a command: delete current picture or undo deletion; (35) tilt de device to select the crop command; (36) passive control of touch force to select the user desired action; (37) passive control of touch force to select dragging command.

Sketching with accelerometers

In his experiments reported in [Rekimoto 96], Rekimoto motivates the use of tilt by the simplicity of sensing motion as variations in angle rather than by changes of position. Levin *et al.* propose an original method to sense positions through accelerometers by using acceleration first- and second-order derivatives (called respectively, *jerk* and *jounce* gestures) [Levin 99]. Their paper describes a fine tuned interaction (14) where physical translations are mapped into system translations as a series of positions. The algorithm is applied to a sketching tool to show the sharpness of the approach. This example illustrates the expressive power of my notation where jerk and jounce are represented with ^{3} and ^{4} exponents syntactic modifiers. It is a continuous interaction technique (absence of clutch to start and stop the interaction). On the other hand, the interaction technique introduces one level of indirection since the gesture acts on a virtual pen (which in turn draws on the canvas).

Rock'n'Scroll

The interaction techniques I have analyzed so far are concerned with foreground interaction only. At the opposite, in [Bartlett 02], Bartlett focuses primarily on background interaction: the system tries to understand user gestures in order to adapt dynamically to context changes such as screen orientation (15) [Bartlett 02]. For foreground interaction, Bartlett mimics familiar gestures to control the orientation of pictures by tilting the device vertically (16), while a horizontal tilt is used to select the next/previous picture by the way of a menu (17). Thus, (17) introduces a level of indirection. The tilt gesture (18) is also proposed for a 3D game to control the movements (position) of an avatar which in turn controls the game, thus introducing a level of indirection. While (15), (16) and (17) are position-controlled, (18) uses a speed control type.

TiltType

TiltType (19) supports text input by combining tilt angles to select characters organized in five position-controlled circular menus multiplexed through the use of five buttons [Partridge 02]. The buttons also serve to trigger the interaction which therefore denotes a clutched interaction technique. In addition, selecting characters from circular menus introduces an indirection with regard to the text entry task.

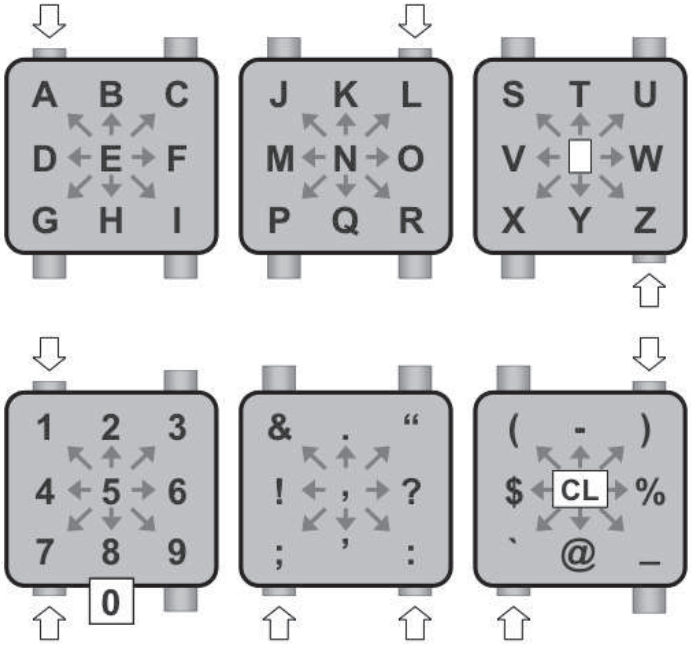


Figure 4.4.: Partridge et al. propose to map a position-based interaction technique drive by accelerometers to a menu containing the most used characters [Partridge 02].

Hinckley's state of the art

As discussed in Chapter 3, Hinckley *et al.* have developed an early classification of interaction techniques for mobile devices [Hinckley 00]. From this taxonomy, they propose a couple of interesting applications that are consistent with my own work by enhancing the background interaction already introduced in Barlett [Bartlett 02]. First, the paper models the screen orientation control by defining the bezels each zone should have in order to prevent from unstable situations; second it applies the same concept to the power management of the device by defining situation where the screen should switch off, letting the user to implicitly select power management options (20).

XWand

XWand allows users to select a device within a multimedia environment by pointing at the device of interest (21) and then to control it with gestures such as tilting the wand to control the volume level (22) *et al.* [Wilson 03]. The paper proposes a selection interaction defined by complex physical gestures that combine translation and rotation. This is why (22) lies at the frontier between translation and rotation. The same role (and

	Context Variable	Description
	TiltAngleLR, TiltAngleFB	The left/right and forward/back tilt angles, in degrees. (sensor reading & transform)
	DisplayOrientation & Refresh	*Flat*, *Portrait*, *LandscapeLeft*, *LandscapeRight*, or *PortraitUpsideDown*. A Refresh event is posted if apps need to update orientation
	HzLR, MagnitudeLR, HzFB, MagnitudeFB	Dominant frequency and magnitude from FFT of tilt angles over the last few seconds
	LookingAt & Duration	If user is looking at the display
	Moving & Duration	If device is moving in any way
	Shaking	If the device is being shaken vigorously
	Walking & Duration	If the user is walking

Table 4.1.: Hinckley classification for Tilt/Accelerometer input devices [Hinckley 00].

indirection level) widgets propose in desktop or mobile devices metaphors, is reproduced by XWand with real world objects thus introducing a level of indirection.

Peephole displays

A peephole is a spatially aware handheld display used as a window on a large (virtual) workspace [Yee 03]. Yee introduces two interaction techniques for this purpose based on accelerometers: (23) uses physical translations to position the peephole over the workspace; (24) enables the selection among different views of the same content through physical translations. Both interactions are position-based and continuous. Whereas (23) directly maps physical translations into the desired position, (24) interprets user gestures to control view changes, thus introducing a level of indirection.

Gestural Authentication

Path is one of the least explored interaction tasks. Even though this task is defined as a composition of orientations and positions, it is unique because it also considers time

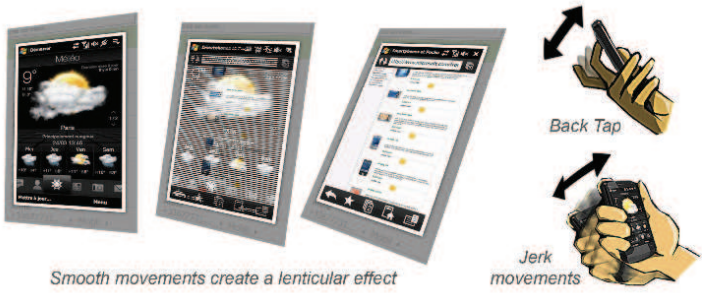


Figure 4.5.: Tapping the back of the device trigger the continuous/discrete mode. In continuous mode application switch is activated by tilting the device. In discrete mode application switch is achieved using *jerk movements* [Roudaut 09].

as a component. The only work classified under this category comes from the studies conducted by Patel *et al.* [Patel 04]. The paper describes a public authentication method driven by a series of well-defined shock gestures applied to the device (25). The series of gestures constitutes the authentication path for a particular user. It is a clutched interaction technique since it is activated by the user on the authentication public terminal. It introduces a level of indirection since the user has to follow instructions proposed by the terminal screen.

Shoogle

Williamson *et al.* develop the idea to use shock gestures to sense a quantity [Williamson 07]. By shaking their mobile phone, users can quantify the number of unread messages, of lost calls, or evaluate battery life (26). Each application of the Shoogle interaction technique is based on a metaphor which then introduces some indirection. Messages are assimilated to balls and the battery charge to a liquid quantity while the container is the device itself. Shoogle uses audio feedbacks correlated to the quantity of balls, or to the amount of liquid contained in the device.

TimeTilt and TapTap

Inspired from real world objects, Roudaut *et al.* propose two TimeTilt interaction techniques based on accelerometers [Roudaut 09]. (27) is a position-controlled interaction technique that enables choosing among different views by rotating the device. The interaction is triggered by tapping the back of the device (28). (29) supports switching

through applications by shocking the device. It does not need to be triggered (29). (27) introduces a level of indirection since it depends on the graphical widgets that renders the different views.

Miscellaneous

Kratz *et al.* focused on feedback implications for gestural interaction techniques [Kratz 09]. They propose a position-controlled gestural menu to select an option through physical gestures (30). I decided to classify this interaction technique as pragmatically direct as the paper does not associate any system command to the proposed gestures. The last interaction technique I analyze is the Apple's shake gesture available on the iPod to skip to the next song (31).

Hinckley's Synaesthesia

A recent work proposed by Hinckley *et al.* studies different accelerometers-driven, touch triggered interaction techniques [Hinckley 11]. Tilt-To-Zoom (32) proposes tilt-to-zoom interaction techniques. The user touches the screen to activate the modality. The tilt of the device backward or forward activates a zooming animation of the screen context at constant speed. Pivot-To-Lock (33) lets users lock the screen orientation by tilting the device. The interaction is triggered by touching of the screen. This interaction is characterized by a missing pragmatic modifier as the finger on the screen acts as the pivot center of the screen rotation. Hold+Shake (34) permits users to select the delete (undo deletion) command on the parameter specified by the finger (the trigger of the interaction). In Tip-To-Select (35) a two finger zooming interaction can enter/exit the crop mode by tilting the device during the interaction itself. The finger acts as a trigger. In Hard-Tap (36) and Hard-Drag (37) the user does not intentionally use the accelerometers by completing a well defined physical gesture. The accelerometers are used to enrich touch driven interactions simulating hard taps of the screen. These interactions are touch triggered. In addition, they are characterized by a pragmatic modifier as they invoke different commands in different contexts.

4.2. Discussion

Figure 4.3 provides an overall picture of more than 30 representative accelerometer-based input interaction techniques. The visual structure of this representation reveals three interesting facts: the absence of scale-based interaction techniques, *the dominance of the Select and Position interaction tasks*, with a *majority for pragmatic indirection*.

The absence of scale-based gestures is likely due to current technological limitations: typically, mobile devices are currently made of rigid material that limits the development of deformation-based interaction techniques. Nevertheless, the scale affine transformation opens the way for future research: in the near future, users will be able to shape their own devices as demonstrated in the early prototypes developed by [Schwesig 04, Hemmert 08]. In addition, it is reasonable to envision twist- and scale-based interactions using accelerometers as input devices to propose a candidate language to perform, among others, the Path interaction task in a simple manner.

As made obvious by Figure 4.3, the centre of gravity is located in the lower part of the taxonomy. Clearly, *most interaction techniques based on accelerometers are concerned by the Select and Position interaction tasks*. Surprisingly, the use of accelerometers for specifying Orientation has not been explored extensively. Therefore, my classification suggests to concentrate research efforts on the development of interaction techniques to support Orientation, as well as Path, Quantity and Text input interaction tasks.

Most proposed accelerometer-based interaction techniques are characterized by *indirect I/O pragmatic connections*. Interestingly, Selection and Position are rarely implemented through pragmatically direct techniques. This is in contrast with mouse-based Foley's Select and Position atomic tasks. It results from such observation that accelerometers-based interaction techniques are not necessarily well suited to conventional WIMP interactors. The taxonomy *brings forward the difference between mouse-based interaction techniques and accelerometer-driven ones*. By construction, accelerometers sense acceleration (i.e. the direction of gravity essentially). These observations suggest to consider Orientation as an atomic task for accelerometers-based gestural interactions.

In addition to the three main observations revealed by Figure 4.3 (i.e. absence of scale, dominance of Select and Position, as well as primacy of indirection), the fine-grained structure of the taxonomy provides researchers and designers with the appropriate apparatus for sound reasoning. An indication of this is that my taxonomic space has allowed us to understand intrinsic and implicit differences even among apparently similar

interaction techniques such as for example between (14) and (23) which both perform positioning in very distinct manner.

From the researcher's point of view, the classification shows a transparent state of the art where each interaction technique is classified without ambiguity. Typically, reference taxonomies such as [Foley 90b] or [Buxton 83] do not consider the role of time (cf. frequency and duration), nor do they cover unconscious interaction (cf. background interaction) and unstructured interaction such as device shaking. In addition, they do not explicitly consider whether an interaction technique is clutched or unclutched introducing ambiguities and mixing up different aspects of human interaction behavior. Hinckley *et al.* recently proposed a taxonomy of accelerometers-driven interaction techniques that consider such parameters, but limiting the case to touch (or none) based triggers. The proposed approach permits the classification of a wider range of interaction techniques that could be presented in future (voice or camera based triggers). Moreover, it considers a fine grain analysis of the implemented user control (cf. the syntactic and pragmatic axes) permitting the unique integration of original interaction techniques such as the one proposed by [Levin 99]. As seen at the end of the previous chapter, the taxonomy proposes a general framework able to consider traditional (WIMP) interactions.

From the designer's point of view, the dimensions of my taxonomy can be used as a framework for decision making. For example, an unclutched interaction technique may be considered for default tasks, while different clutched interaction techniques can be multiplexed through the use of standard or ad-hoc widgets. By proposing at least an interaction technique for each of the proposed task while designing an application, designers will be able to offer a complete and uniform user experience similar to the WIMP one. Furthermore, designers can predict the difficulties that final users will encounter by analyzing the pragmatic and syntactic modifiers that characterize the interaction techniques they envision. Thus, they will be able to choose interaction techniques that best suit the targeted representative users (novice, intermediate, expert). In addition, they should be able to choose and implement the interaction techniques that best suit the targeted representative users (novice, intermediate, expert) as the classification proposes an overview of the learning difficulties in terms of syntactic and pragmatic modifiers. Good research and development directions will be both towards the creation of widgets that are able to transform direct interactions in their more complex counterparts and toward the definition of the elementary interactions to base the development on.

4.2.1. My Definitions

The previous section presented a state of the art on the acceleration-based interaction techniques and the taxonomies proposed to organize the human-computer dialogues and the associated input device. Several point of views have been considered and dissimilar definitions presented. Here the goal is to retrieve some principal concepts future section will be based on. The definitions contained in this section doesn't want to be universal, but need to be valid for the scope of this work.

Interaction Technique

As introduced by [Card 91], the interaction between human and machine is a dialogue spoken using an artificial language. Humans specify the words by the means of the input devices and composes the sentences using previously defined interaction techniques.

An interaction technique needs to live on a well defined language to make the final user understand the proposed metaphor [Shneiderman 87] and to let him imagine the state of the system and the next step in the interaction itself. The Model of Human-Processor [Liu 06] state the importance of the perception during an interaction. Beaudouin-Lafon [Beaudouin-Lafon 00] notices the importance of the feedback as a key piece of the interaction while Coutaz highlight the context [Coutaz 05]. In the context of accelerometers-base interaction techniques, Kratz et al. [Kratz 09] demonstrate how the feedback proposed to the user influences the execution of real world gestures by improving the algorithm performances.

Tilt

Several definitions have been proposed and misinterpreted around the meaning of the *Tilt* interaction on mobile devices. For the scope of this work I accepted the simplest vocabulary definition interpreting it as a synonym of lean, that is *to incline or bend from a vertical position*. The word itself doesn't imply any syntactical characterization of the interaction. As such a tilt-based interaction technique can be characterized by a position, speed or acceleration control. Can be based on one, two or three axis lean. Is associated to any kind of semantical operation at system level, i.e. selection, position, or any of the possibilities available.

Shock

According to the definition given by the proposed taxonomy, I will consider the *Shock* as an unstructured repetition of an affine transformation. A shock is an atomic gesture. From a physical point of view it can be a sequence of translations going on a direction and than on the opposite one. I retain that from the user perspective, any shock gesture (such as the Shake) is considered as an atomic gesture (after all, apart from the duration property, is it conceivable an "half a shake"?).

4.3. Synthesis

I applied my taxonomy to accelerometers-based interaction techniques. I demonstrated the flexibility of my approach, still I enabled a user-center discussion around gestural interaction techniques. The syntactic and pragmatic modifiers of my classification space provide a sound predictive measure for the learning curve users have to go over when approaching a new interaction technique. In the next section I will extend my approach to multimodal/crossmodal gestural interaction techniques. I will present my experiments, my goals and my vector-based, hierarchically enable a gestural framework I used to implement gestural menus.

Part III.

Composing Touch and Tilt

Chapter 5.

TouchOver

5.1. Introduction

Touch-enabled devices, especially handheld ones, propose interactions often described as more "natural" to the user. However, a limitation of this kind of input devices is hardly ever acknowledged: it lacks a passive state. In other words, the touch always performs an action (tapping, dragging, etc.) which starts as soon as the finger touches the screen and ends only when the finger leaves the screen. The "natural" deictic property of the finger [Karam 05] is thus in fact often not respected as the finger acts on the interface as soon as it is in contact with the screen. This very same usability problem has already been observed in the field of gaze-enabled interactions: the *Midas touch* term was coined by Jacob to denote unintentional selections for eye-gaze input: "Everywhere you look, another command is activated; you cannot look anywhere without issuing a command" [Jacob 03].

In fact, touch input do have a passive state: it is when no finger is in contact with the screen. But no tracking can be performed while being in this state, so no input is available to the system. The missing state of touch devices is in fact a passive *tracking* state, i.e., a state in which a position is given to the system but no action is engaged. Having such a state matters for various reasons: it allows to multiplex interaction techniques, and it would allow a better precision for target selection.

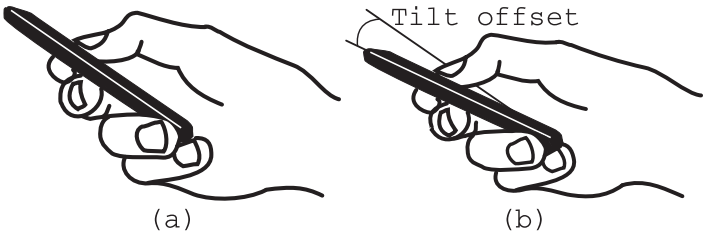


Figure 5.1.: With touchOver, users can switch between two interaction states (a) and (b) with a simple tilt of the device while still interacting with their finger. This permits for example interactions such as hovering, dragging, feedforward enabled techniques, visual and eye-free interface exploration, and selection precision improvement.

5.1.1. The Need for Multiplexing Interaction Techniques

As handheld devices evolve technically, more complex interactions are required to take full advantage of the technology. For example, the number of features offered by the iPhone home screen has increased since its first release in 2007. It now includes icons grouping, a multi-task dock, and a search page. In order to offer extra interactions, current smartphone operating systems use time-based touches and/or modes. For example, where legacy desktop web browsers use the *mouseover* state to display a web page link address, smartphones web browsers like Mobile Safari or Android's web browser, use a *touch and hold* to display a pop-up window with the link address along with buttons for potential actions. Another example is a *touch and hold* on an application icon that enables the edit mode of the iPhone's home screen. In this mode, the user can drag and delete application icons. The physical home button of the device must be pressed to return to the nominal pointing and scrolling mode. Although attractive at first sight, these techniques have some drawbacks and break the interaction flow.

5.1.2. Precision of Selection

Another more subtle problem is the consequence of having a single tracking state on touch screens: the lack of precision for selection tasks. In fact, most touch devices sense the fingers a little bit before an actual touch occurs: as the finger approaches the sensitive surface, the system immediately starts tracking the finger. At the other end, the user's finger continues to be tracked after it is lifted from the screen. This results in the system acquiring extra motion events, and misplacing *press* and *release* events.

While this imprecision is negligible for sufficiently wide targets, it limits the efficiency of precise pointing techniques.

5.1.3. Introducing a Passive Tracking State

Based on these observations, we propose a novel approach for touch-based, accelerometer-enabled, handheld devices: the TouchOver technique (depicted on Figure 5.1). My approach:

- provides a three-state transition input model, thus re-opens the opportunity for richer interaction techniques through the intermediate *on-over* state;
- reduces the *Midas touch* effect by using two orthogonal modalities to transition between states, thus improves position and selection precision; and
- encompasses the traditional two-state model of touch, thus remains compatible with most existing interaction and precision-improvement techniques.

In the next section, I discuss the theoretical background that motivates a three-state input model from which TouchOver is derived and I review in details the state of the art of touch-based precision improvement techniques and handheld devices tilt-based input interactions. I then describe the TouchOver technique and its design. The controlled experiment I have conducted to compare selection precision and the results I found are then presented. Before giving my conclusion, the discussion will drive us through the analysis of my hypothesis with respect to the experimental results.

5.2. Observations

5.2.1. State Models of Input Devices

Finite state machines have been used by Buxton *et al.* [Buxton 90] as a simple and effective model to characterize and compare input devices in the context of direct manipulation. Figure 5.2 depicts the state machine of the mouse input device according this formalism: it consists in two states, *tracking* (labelled "state 1") and *dragging* ("state 2"), the transitions between them occurring on the *button up* and *button down* events. In both states, the motion of the device is tracked.

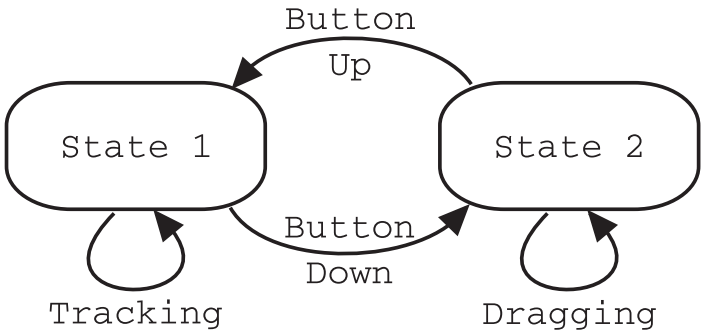


Figure 5.2.: Mouse input state machine (reproduced from [Buxton 90]).

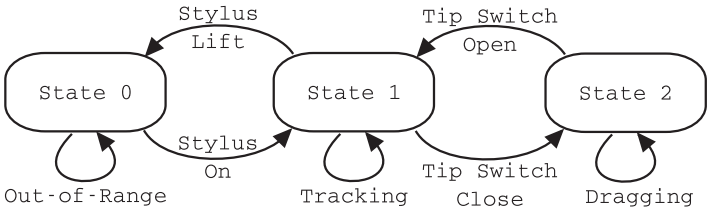


Figure 5.3.: Stylus on graphics tablet input state machine (reproduced from [Buxton 90]).

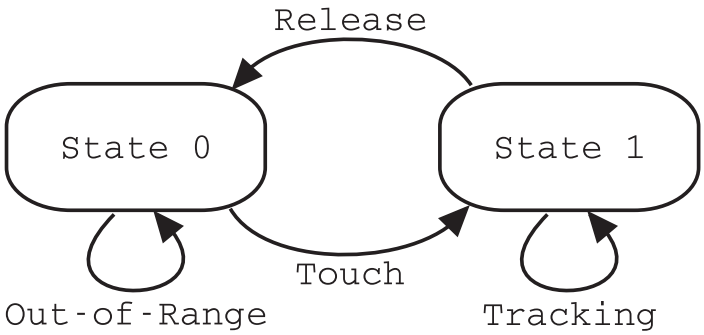


Figure 5.4.: Touch-screen input state machine (reproduced from [Buxton 90]).

Figure 5.3 gives the state machine for a stylus on a graphics tablet: It extends the mouse state machine with a supplemental *out of range* state ("state 0") which is active when the stylus is not present on the tablet. The *tracking* and *dragging* states are distinguished by the stylus tip switch. As with the mouse, the stylus is tracked in those two states

With touch-screens, the finger is either on the sensing surface or out-of-range for the tracking system. As Figure 5.4 shows, this results in an input state machine with only two states: the *out-of-range* state ("state 0") where the system is unaware of the finger's motion; and the *tracking* state ("state 1") where the system tracks the finger's motion. Transitions between those states are operated when the finger lands on, or lifts from, the sensing surface. As discussed by Buxton *et al.*, this differs significantly from the mouse

although both input behaviors can be modeled as a two-state machine [Buxton 90]. As shown in Figure 5.2, the mouse is never out of range for the tracking system. The main difference is that for the touch input, the fingers are tracked only in one of the two states of the machine.

In many interaction techniques, when the *dragging* state is present, the *tracking* state is exploited to provide users with additional information such as feedback and feedforward to support user's subsequent actions. On the other hand, the applicability and usability of the two-state model has been demonstrated by its rapid spread in commercial products, by end-users' acceptance, and by the interest of the HCI community. But, as seen in the Introduction, modes, quasi-modes, and timeouts have been introduced to extend this simplistic two-state input machine.

One of the popular technique used to introduce a supplemental mode is the aforementioned *touch and hold* (or long-press) interaction. It has however the drawback of limiting the speed of interaction as the user has to wait for the system to switch to the secondary mode. The timeout duration is a trade-off between the speed of interaction and the risk of misundersting the user's movements. Furthermore, the most desirable trade-off varies depending on the user's expertise and the context of use.

5.2.2. Improving Precision of Selection on Touch Devices

Difficulties with precise interaction on touch-screen mobile devices have been acknowledged and addressed before. A possible solution to the problem of precise selection with fingers is to align the target size with the "resolution" of the fingers. For instance, Parhi *et al.* recommend that the minimum size for thumb input should be "9.2 mm for single-target tasks and 9.6 mm for multi-target tasks" [Parhi 06]. When considering small touch-screens, this constraint is quite strong as this size can represent up to one tenth of the whole screen real-estate. Other approaches to overcome the lack of precision include: avoidance of target occlusion by fingers, hands, or arms; or alteration of the control-to-display ratio.

In the first case, the techniques improve precision by avoiding occlusion through visual augmentation [Vogel 07, Yatani 08] or technical innovation [Wigdor 07]. The control-to-display ratio manipulation approach, used to improve touch screen precision, has been explored by various interaction techniques [Albinsson 03, Olwal 08, Olwal 03, Blanch 04, Roudaut 08]. For some, the interaction complexity of the proposed solution does not fit

with mobile devices requirements. For others, the complexity of the architectural design makes it difficult to apply in practice. However, these approaches suffer from a common underlying problem: the aforementioned *Midas touch* effect.

The fine characterization of the problem (i.e., that the finger is tracked a bit more than expected) was done by Potter *et al.* by analyzing positioning and selection on touch-screen devices [Potter 88]. Their study consider positioning and selection as two different tasks that users need to accomplish separately. Nevertheless, the proposed solution strategies (select either while landing on the screen or when lifting from it) in practice merged selection and positioning. Precision on land-on selection based touch-screen have been later discussed by Sears *et al.* [Sears 91]. Benko *et al.* [Benko 06] recently build upon this work to propose the *SimPress* clicking technique aiming to let users explicitly perform selection as proposed in my *TouchOver*. However, *SimPress* does not use a complementary multimodal interaction as *TouchOver* does, and thus suffers from the *Midas touch* effect.

5.2.3. Multimodal Techniques

The error caused by the erroneous tracking of the finger is unpredictable and variable among users and situations. As explained below, my solution proposes to decouple positioning and selection tasks, by using two different modalities, namely the touch for specifying a position, and the accelerometers to specify a mode switch. As seen the use of the finger as a pointing device identifying a point in space is naturally justified by its native deictic properties [Karam 05] and by the direct manipulation paradigm. While the use of an accelerometers-based spatial gestures [Ekman 72] is motivated by the device properties of sensing real world actions.

The use of accelerometers sensed actions as a trigger for a specified action has been addressed before. Accelerometers have been extensively used to accomplish a well-defined command in different contexts. Roudaut's *TimeTilt* proposes two interaction modalities to switch application context on handheld devices [Roudaut 10]. Both modalities let users choose the context through spatial gestures tilt or translation of the device respectively. Indeed, the first modality is activated by a tap on the back of the device itself. Williamson's *Shoogle* activates an audio feedback to communicate information on the state of the device [Williamson 07]. Fitzmaurice *et al.* used the tilt of the device to control pie menu selection [Fitzmaurice 93], while Rekimoto used the gesture to control the selection on a linear menu [Rekimoto 96].

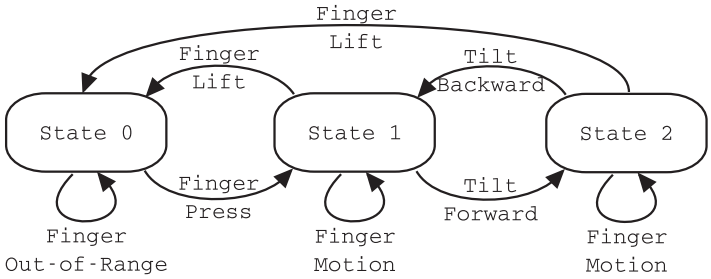


Figure 5.5.: The TouchOver input state machine.

5.3. The TouchOver Technique

The absence of a *dragging* state for touch-screens implies that Foley's two elementary tasks [Foley 90b], positioning and selection, are bundled together. The positioning task involves specifying a 1D, 2D or 3D position in the application specific coordinates system. The selection task involves the validation of the current position to select the underlying GUI interactor. These tasks rely on each other, in particular, the selection task relies on positioning. As made explicit by Figure 5.4, the interaction model of touch-screen based interactions consists in using the last position tracked by the system as the candidate for the selection task. Thus, the selection command is triggered when the user lifts his/her finger from the screen. In other words, when users start specifying a position, they start the selection process. When they finish specifying a position, they confirm the selection. As opposed to the mouse interaction model, the two interaction tasks are not controlled separately, which in turn may cause the *Midas touch* effect.

For the design and implementation of *TouchOver*, a three-state transition input model (shown in Figure 5.5) is exploited to decouple positioning from selection while maintaining a fluid transition between the two: positioning is achieved through the absolute pointing of the finger on the touch-screen while selection is performed by validating the current position by a gentle tilt of the device sensed by accelerometers (Figure 5.1). As highlighted by Buxton *et al.*, a three-state graphical input model "can characterize [...] many of the demands of interactive transactions, and many of the capabilities of input transducers" [Buxton 90]. *TouchOver* exploits the three-state machine to decouple positioning from selection, but supports a smooth and continuous transition between these two interdependent tasks. Here, decoupling allows the user to control each elementary task explicitly, making it possible to freely compose and link them together. Continuity is ensured by assigning two complementary interaction modalities [Coutaz 95], seamlessly integrated, each independently implementing one

elementary task. *TouchOver* uses direct touch input only for positioning, and gesture for the selection task.

Gesture recognition is supported on recent handheld devices as they come with an increasing variety of physical sensors such as accelerometers, digital compasses and gyroscopes. *TouchOver* uses a gentle and smooth tilt forward to send the press-like event, and a smooth tilt in the other direction (bringing back the device to its original position) to send a release-like event as illustrated in Figure 5.1. This gesture is combined with touch input to create a three-state input state machine (Figure 5.5). Note that this machine is very similar to that of the graphics tablets with stylus (Figure 5.3). This brings more interactive possibilities to touch-screen based accelerometer-enabled handheld devices. An inverse configuration of the gestures directions (smooth tilt backward to send the press-like event, and a smooth tilt forward for release-like event) has been considered as an alternative. Preliminary tests showed this configuration being more error prone. The easiness of the wrist to execute backward tilt gesture made users exaggerate the movement with a consequent loss of screen visibility.

The chosen selection gesture has the following interesting characteristics. First, it is easy to learn and to remember as it introduces a kinesthetic mode [Sellen 90]. Then, it is smooth rather than impulsive [Baglioni 09] and of a relatively small amplitude compatible with the wrist capacities. As a result, users can keep their field of vision focused on the screen while performing the gesture. It is easy and relatively fast to perform, and it does not rely on timeout like *Long-Press*. This will be further discussed with the experimental results reported below. As a smooth gesture, it also generates less strain than an impulsive one. Due to the low amplitude of the gesture, the finger ease of motion is similar in both states. Furthermore, as a complementary multimodal interaction [Coutaz 95], the hand gesture confirms the finger choice completing and emphasizing its movement without disrupting the user's working flow [Sellen 90].

I have also applied the design principles of *TouchOver* to a timeout-based selection. While using the same positioning interaction technique, the selection is triggered by a timeout fired when the user stops moving on the touch-screen. This is discussed in more detail with the results of my experiments.

5.4. Experimental evaluation

I conducted two controlled experiments with three different validation techniques. For all three techniques, positioning was performed with the finger in contact with the touch-screen. The three validation techniques were:

Take-Off where validation is performed on finger lift from the screen. This is the touch pointing interaction commonly used when position adjustments are required as for smartphones soft keyboards.

TouchOver where validation is performed by tilting forward the device while the finger is touching the screen. The validation is performed when the orientation angle along the axis parallel to the width of the screen is more than 11 degrees different from the angle when the user pressed its finger. An audio feedback, together with a graphical deformation emphasizing the perspective, inform the user that the tilt is sufficient and that the system has recorded the validation.

Long-Press where validation is performed when the user keeps his/her finger still for 1 second. An audio feedback, together with a graphical deformation simulating the press-state of the whole interface, inform the user that the timeout has elapsed and that the system has recorded the validation.

While *Take-Off* provides a kinesthetic feedback of the validation, the other two techniques requires a feedback to inform the user that the system acknowledged the validation.

Based on my rationale and pilot testing, I hypothesized that:

- (H1) *Long-Press* outperforms *Take-Off* in terms of precision at the expense of increased task time due to the timeout the user must wait for.
- (H2) *TouchOver* outperforms *Take-Off* in terms of precision at the expense of increased task time due to an extra user's action.
- (H3) *Long-Press* outperforms *TouchOver* in terms of precision but with less difference than for *Take-Off*.

In addition to testing these three hypotheses, my goal was also to get indicators about the impact of the extra action that *TouchOver* requires in terms of strain and validation time.

5.4.1. Apparatus

The experiments were conducted on the iPod Touch 4th generation 8GB running iOS4.2.1. The screen is 3.5 inches wide with 960 x 640 pixels (resolution 326 dpi). When using one finger, the minimal touch motion reported by the system during the pilot and formal experiments was 0.5 point (0.18 mm for a resolution of 144 dpi) or 2.25 pixels. I used such a device to get high resolution screen and touch sensor to minimise the limitation of the interaction due to the output display and the input sensor resolution. As opposed to usual subpixel accuracy of pointing devices, with this device, the touch input is less precise than the screen.

5.4.2. Participants

Eighteen right-handed unpaid volunteers (2 female), ranging in age from 21 to 33 years were recruited from students and staff of my university and university unrelated persons. All but one participants had prior experience with touch-screen based handheld device among whom 11 used it on a daily basis.

5.4.3. Procedure

Participants were first explained the three techniques and could test and learn them with a sample application. Then, they had to perform a first experiment that would focus on validation precision, followed by a second experiment centered on validation time. Participants were instructed to perform both experiments with their dominant hand while standing-up still. For each technique, they were asked to fill in a qualitative questionnaire just after the trials.

For *the precision experiment*, participants were asked to reach a position on a one-dimension vertical axis as precisely as possible and to validate it (Figure 5.6) for each technique. This task is similar to setting the thumb of a vertical slider at a very precise position. The position to reach was figured by a horizontal dashed line displayed in the middle of the screen. A second horizontal dashed line was displayed at a distance from the position to reach proportionally to the distance between the on-screen finger's position and the position to reach. This second line both solves the occlusion problem and adds some control-to-display gain. Overcoming the occlusion problem is necessary for precise positioning on touch-screens. I found during pilot testing that reaching a

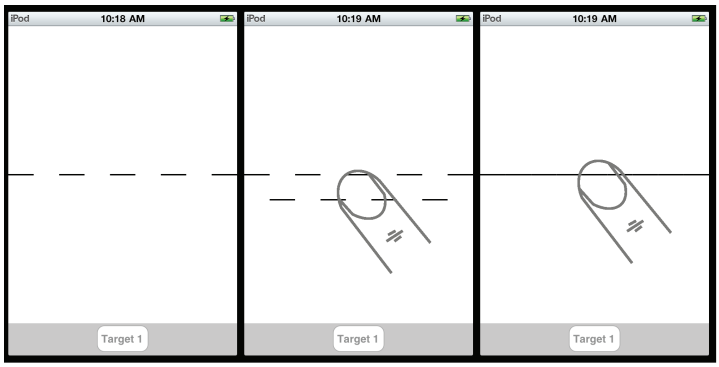


Figure 5.6.: The user interface for the precision experiment: the target line (left); the thumb approaching the target (center); the thumb is on the target (right).

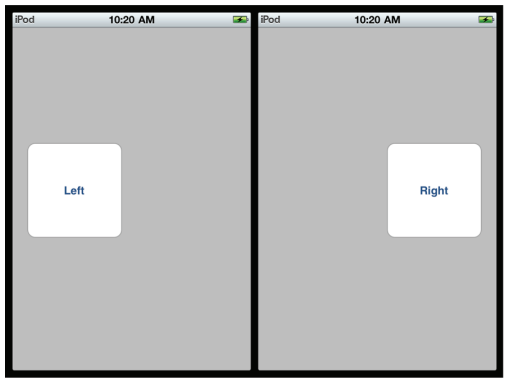


Figure 5.7.: Speed experiment application interface: left the first target; right the second target.

position with such precision (0.18 mm) without display zoom and control-to-display ratio was really demanding for user's vision. As we aimed at testing the limitations due to the input, not the output, I zoomed the dashed line by 4 compared to the motor space, for a resulting line width of 2 points (0.7 mm) and add a control-to-display ratio of 5.

For *the validation time experiment*, participants were asked to select buttons as quickly as possible with the presented validation technique. They had to select two buttons consecutively (Figure 5.7). For this experiment, participants benefited from learning from the previous experiment.

5.4.4. Design

A repeated measures within-participants design was used. Presentation order of *Techniques* was counter-balanced across participants. The 6 permutations of the 3 *Techniques* presentation order were each repeated for 3 participants.

In the precision experiment, the initial goal was to measure the influence of *Technique* on the error rate and the error distance. Yet, I kept a precise positioning phase prior validation to evaluate the techniques in this particular context: during pilot testing, I observed that the finger can be placed differently after a positioning adjustment. Furthermore, this difference in position can affect the finger lift movement as well as its stability while tilting the device or holding a position for a certain amount of time.

For the precision experiment, the independent variables were *Technique* (*Take-Off*, *TouchOver*, *Long-Press*), *Block* (first, second, third) and *Presentation order* (the 6 permutations). The measured variables were *Distance* and *Errors*. In summary, the experimental design was: 3 *Techniques* (*Take-Off*, *TouchOver*, *Long-Press*) x 3 *Blocks* x 15 trials per *Block* = 135 data points per participant.

In the validation time experiment, I measured time from the moment the finger landed on the target button to the moment the validation was performed. For *Long-Press*, this measure is a measure of the timeout duration and the system ability to perform a precise timeout and eventually its capacity to detect finger stillness. For this reason, *Long-Press* was not included in the validation time analysis. Nevertheless, I performed the speed experiment for *Long-Press* to keep the same experimental sequence for all three techniques.

For the validation time experiment, the independent variables were *Technique* (*Take-Off*, *TouchOver*), *Target* (left and right) and *Presentation order* (the 6 permutations). The measured variable was validation *Duration*, the time spent between the finger press event and the target validation. In summary, the experimental design was: 2 *Techniques* (*Take-Off*, *TouchOver*) x 2 *Targets* x 5 trials per *Target* = 10 data points per participant.

5.4.5. Results

Error Rate

For the precision experiment, I considered the error rate while validating the one dimension position whose size (0.18mm, or 0.5 point) is that of the resolution of the touch sensor system. For each of the validation techniques, I performed a Pearson's Chi-squared independence test between whether the target was acquired successfully or not, and the 3 *Blocks*. I did not found any significant dependency between target acquisition and *Blocks*. Thus, there is no evidence of either learning or tiring effects for any of the techniques.

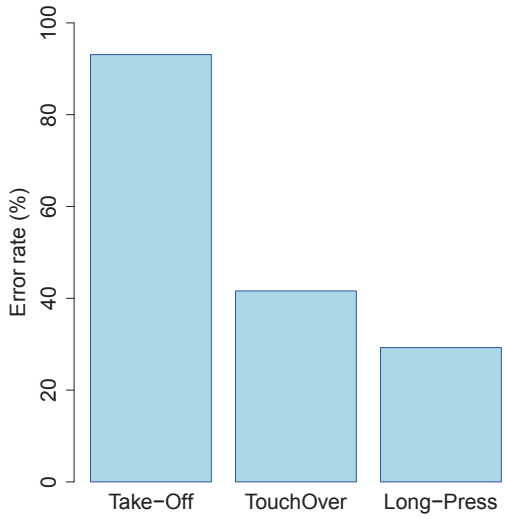


Figure 5.8.: Percentage of errors by validation techniques

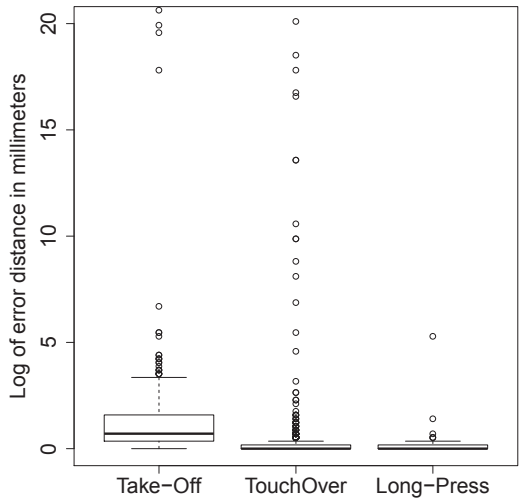


Figure 5.9.: Boxplot of error distance in millimeters grouped by validation technique

I performed a Pearson's Chi-squared independence test between whether the target was successfully acquired or not, and the 3 *Techniques*. This test shows a significant dependence (X-squared = 749.16, $p < .0001$). The Cramer's V statistics validates the intensity of the relation ($V = 0.5$). The overall error rate percentage is 55%. This percentage is higher for *Take-Off* (93%) than for *TouchOver* (42%) and *Long-Press* (29%) (Figure 5.8). This supports my three hypotheses, H1, H2, H3.

Error Distance

During the precision experiment, I measured the distance by which the position to reach was missed with a distance equal to zero when the position was successfully validated. As the distribution of *Distance* grouped by *Technique* (Figure 5.9) shows, each *Technique* distribution presents outliers. *TouchOver* presents much more outliers than the two other techniques. From what was observed during the trials, this can be explained by the reliability of the basic gesture recognition I implemented for the experiment. I believe that the number of missunderstood gestures can greatly be decreased with a more robust gesture recognizer. Yet such system missunderstanding cannot be completely avoided and would increase in real mobile usage conditions. So, all data were kept for the statistical analysis.

To correct from absolute error distance right skewness and outliers, I used the median error distance of aggregated repetition for each participant. I performed a 3 x 3 x 6 (*Technique* x *Block* x *Presentation order*) within subjects analysis of variance on median absolute error distance. Significant main effect was found for *Technique* ($F_{2,161} = 97.8102$, $p < .0001$). Neither *Block* nor *Presentation order* were found significant. Furthermore *Technique* x *Block* and *Technique* x *Presentation order* interactions were not found significant either. Post hoc Tukey multiple means comparison test confirmed that *TouchOver* and *Long-Press* are more precise than *Take-Off*. This multiple means comparison did not find any significant difference between *TouchOver* and *Long-Press*. These results support hypothesis H3.

Validation Time

During the validation time experiment, I measured time from the moment the finger landed on the target button to the moment the validation was performed. Figure 5.10 shows validation time distribution for *Take-Off* and *TouchOver*.

I performed a 2 x 2 x 6 (*Technique* x *Target* x *Presentation order*) within subjects analysis of variance on median validation time of aggregated repetitions for each participant. Significant main effect was found for *Technique* ($F_{1,71} = 47.8942$, $p < .0001$). *Presentation order* effect was found significant, though with a p-value of 0.067 ($F_{5,71} = 2.1953$). *Target*, *Technique* x *Target* interaction, and *Technique* x *Presentation order* had no significant effect on validation time indicating that the design of the two-target experiment was appropriate. Post hoc means comparison test confirmed that *Take-Off*

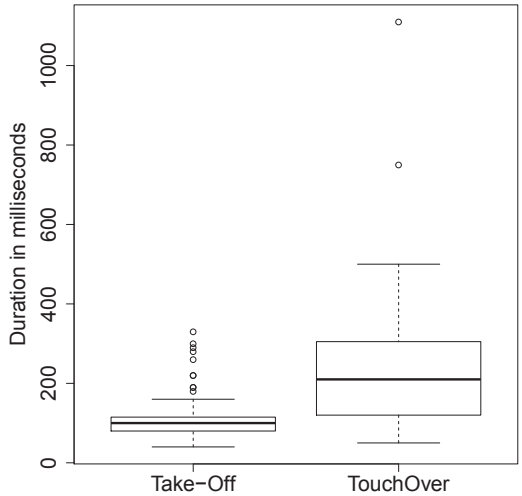


Figure 5.10.: Boxplot of validation duration for *Take-Off* and *TouchOver*

is faster than *TouchOver* (T = -11.7233, p < .0001). The mean validation time was 106 ms for *Take-Off* and 213 ms for *TouchOver* with a 95% confidence interval means difference ranging from 76 ms to 138 ms. This difference can be explained by the extra action required by *TouchOver* to perform the validation. It provides some measurement of the cost of this extra action in terms of task completion time.

Subjective preference

Participants were asked in a questionnaire whether or not they felt any physical tensions in either thumb, hand, wrist or arm during the experiment. From the 18 participants, 4 said they felt physical tensions with *Take-Off*, 6 with *TouchOver*, and 7 with *Long-Press*. Among these participants, three said they felt physical tensions for all three techniques. This suggests that *TouchOver* and *Long-Press* sound similar in perceived physical strain and both seem a bit more stressful than *Take-Off*. Yet I cannot conclude on any strong difference between the techniques. This is comforted by the fact that no significant tiring effect was found during the precision experiment.

Participants were also asked to grade the three techniques for both the speed experiment and the precision experiment tasks. For the speed experiment task there is a trend to find *Take-Off* more efficient and satisfying than the two other techniques (Figure 5.12). For this task, *TouchOver* receives little better notes than *Long-Press*. For the precision

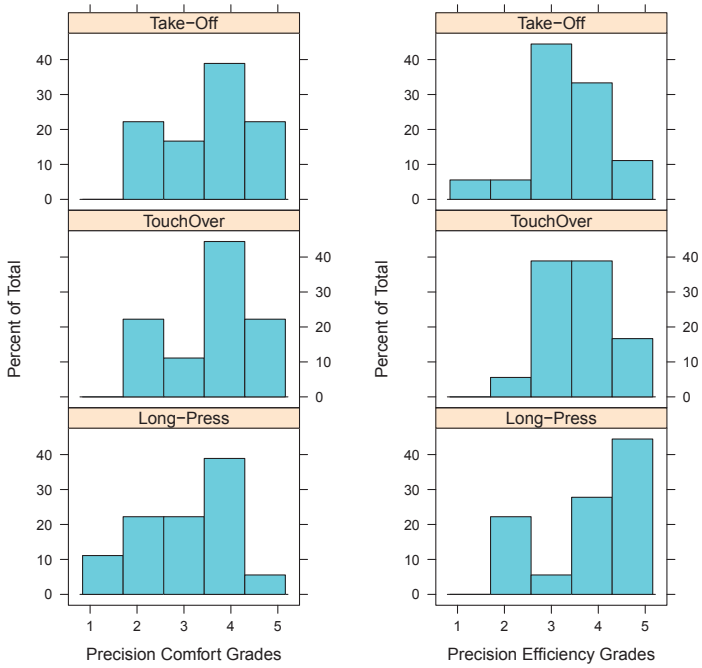


Figure 5.11.: Comfort and efficiency grades for each technique for the precision experiment

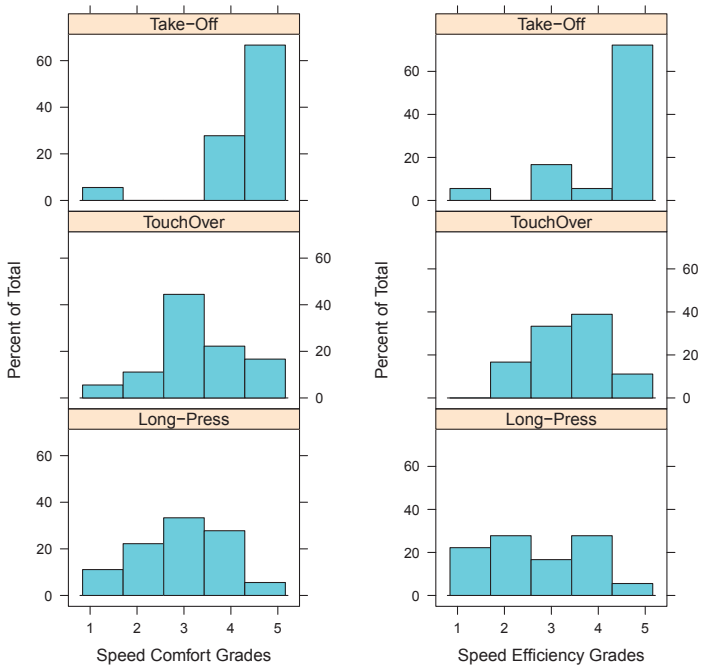


Figure 5.12.: Comfort and efficiency grades for each technique for the speed experiment

experiment task there is a trend to find *Long-Press* more efficient than the two other techniques (Figure 5.11).

5.5. Discussion

My experimental results support my hypothesis, which I based on my technique rationales and design. Like *Long-Press*, *TouchOver* improves positioning validation precision compared to *Take-Off*. Yet, *TouchOver* is not as precise as *Long-Press*. During the experiments, I observed that in some cases, participants used their thumb and fingers along with their wrist to perform the tilt offset of the device. In such cases, user's thumb contact surface changed during the gesture introducing error during validation. As shown by the experimental results, even with this limitation, *TouchOver* still remains of interest for precise positioning validation. I believe that an appropriate feedback along with users' habits and expertise can moderate such gestures and encourage users to use a wrist gesture to tilt the device while keeping their fingers still.

During the experiment, when users were explained *TouchOver*, they understood it and learned how to use it quickly. This is comforted by the fact that I did not find signification indication of learning effect. Nevertheless, some smartphone-experienced users were tempted to shake the device instead of performing a gentle tilt to perform the validation. This can be explained by the recent introduction of shake-controlled commands in commercial products. Indeed, the mainstream experience they provide influences users' expectation on physical gesture based interaction techniques with handheld devices. Again, appropriate feedback and users' habits should minimize such behaviors.

Although my implementation of *TouchOver* was based on a simple gesture recognition, it performed well in a controlled environment. Gesture recognition robustness can still be enhanced by taking advantage of more appropriate sensors like gyroscopes and implementing more sophisticated algorithms. Still, any gesture recognition will suffer from ambiguities. Yet I believe that again an appropriate feedback and users' habits and expertise can also reduce the number of recognition ambiguities.

TouchOver offers a promising trade-off, bringing an extra passive tracking state to handheld devices input and increasing positioning precision at the expense of an extra action. While with *Long-Press*, users need to wait for a timeout to perform the validation, with *TouchOver* they actively control the state transition. Furthermore, experimental

results indicates an affordable cost of *TouchOver* in terms of task duration and physical strain

For common handheld devices interactions like concurrent pointing and scrolling, *TouchOver* is of no particular interest due to the extra user's action it involves. Yet, *TouchOver* allows for richer one finger-based interactions like for example concurrent support for pointing, scrolling and dragging with no need for modes.

5.6. Synthesis

I presented *TouchOver*, a complementary multimodal input for one hand interactions on touch-screen based accelerometers-enabled handheld devices. *TouchOver* offers a three-state model input similar to the stylus tablet input with two states where the system tracks finger's motion, thus adding a passive tracking state to the touch input. This creates new opportunities for handheld device interaction techniques like *on-over* interactions, feedforward, or visual and eye-free user interface exploration.

Furthermore, when positioning validation is performed on the tilt gesture transition rather than on finger press or lift, positioning tasks gain in precision at the expense of an extra action. My evaluation of *TouchOver* in a controlled environment shows an encouraging trade-off between *Take-Off* and *Long-Press*. Indeed it improves positioning precision at an affordable cost in terms of task duration and physical strain. Existing precision improvement techniques can benefit from this gain in precision.

Part IV.

Gestures-Driven Menus

Chapter 6.

Characterizing Gestural Menus

The emergence of multipurpose electronic devices has amplified the necessity for creating novel input devices and interaction techniques to support an increasing number of functions and commands. These include the combination of physical switches, special purpose functions keys, software modes and quasimodes as well as general purpose widgets such as soft-buttons, scroll-bars, lists, and menus.

Among the widgets currently available for user interface design, menus play a key role. They support a fundamental and frequent human task: that of making choices. By contrast with command styles, menus present the possible choices (either graphically or vocally), and only the choices that are semantically valid in the current system state. *Through the menus, all possible actions can be made visible and, therefore, easily discoverable* [Norman 10]. They are an *attractive alternative* [to keyboard command strokes] *because they can eliminate training and memorization of complex command sequences* [Shneiderman 87]. When designed carefully, menus shorten learning, provide a clear structure to decision-making, support exploration, reduce errors, and may be appealing to expert users when they include high-speed interaction shortcuts [Shneiderman 87].

Figure 6.1.: Menus declinations. From left: the Wavelet menu for touch-enabled mobile devices [Francone 09]; Polymorphic Menu developed in the NOMAD project [http://iihm.imag.fr/contract/nomad/], MTM (Menu MultiTouch) for multi-points tabletop [Bailly 09]; Shadow Reaching : target selection using shadows [Shoemaker 07].

Because of their key role in user interface design, menus have been investigated since the early eighties. As illustrated in Figure 6.1, they have evolved as the technology has continued to bring in additional constraints such as small screens, but also new opportunities such as the use of gesture and large interactive surfaces. In this chapter, I am concerned with graphics gestural menus for handheld devices. Before going into the detailed study of gestural menus, I propose to address the question "What is a menu?".

6.1. What is a Menu?

Existing definitions or modeling techniques for the concept of menu make reference to a parent class called a "widget" [Swick 88], an "instrument" or "meta-instrument" [Bailly 09], an "interactor" [Coutaz 93], or even a "technique" [Nancel 09] and "interaction style" [Shneiderman 87]. This diversity demonstrates that the status of a menu as a subclass of a more general concept is unclear. In this section, I analyze the existing definitions for menus, then look into the parent classes they refer to, and close with my own definitions.

6.1.1. Existing definitions for Menus

The Handbook of Human-Computer Interaction [Helander 97] concentrates on the semantic function of menus and on the interaction feedback they produce. It defines a menu as *a set of options displayed on the screen where the selection and execution of one (or more) of the options result in a change of the state of the interface.* In turn, *menu selection is a mechanism for user to indicate their choices; The characteristics of menu selection are that: (1) the interaction is, in part, guided by the computer; (2) the user does not have to recall commands from memory and (3) user response input is generally straight forward.* According to this definition, a menu is necessarily graphical, and a palette is also a menu.

More recently, the transitory nature of menus has been considered where menus are characterized as *transient* [Jakobsen 07] and *quasimodal* [Raskin 00] widgets. *Quasimodes* are modes that are preserved only through some action maintained by the user, such as pressing the shift key. They have the advantage of being a mode without forcing users to remember the actual application state.

Nancel [Nancel 09] summarizes the existing definitions viewing a menu as *a technique permitting the choice of an item among a predefined totality, that can have a hierarchical structure, offering a transient visualization of possible choices during the interaction.* Again, according to this definition, a menu is necessarily graphical, which excludes vocal menus.

Perhaps, the most precise definition for menus has been provided by Bailly [Bailly 09] where a menu is a *meta-instrument* (also referred to as an *interactor* in his thesis) that satisfies the following five properties:

1. It supports the selection of one item within a finite set of options.
2. It minimizes mental efforts by presenting the set of options to the user (favors recognition as opposed to recall).
3. It presents the options as a semantically and spatially meaningful structure that is supposed to match human goals.
4. It is transient in the sense that it is perceivable to the user only during its interaction with the user.
5. It is quasimodal since it defines a local context for system interpretation that is maintained explicitly by the user until the interaction ends.

From there, Bailly introduces a unified vocabulary related to menus:

A *menu system* is a set of menus tied together.

A *menu technique* denotes the *interaction technique* [Appert 04] associated to a menu or to a menu system.

The *current menu* is the menu of a menu system the user is currently interacting with.

A *submenu* is a menu that is accessible from an item of the current menu.

A *super-menu* or *parent-menu* is one of the menus that has made it possible to access the current menu.

Interestingly, Bailly introduces a consistent set of concepts (i.e. menu system, menu technique, current, super- and sub-menus) to analyze menus, their interactive behavior and their organizations. However his reference to a "spatially meaningful structure" indicates that vocal menus (whose structure is expressed with temporal relations) are not covered by his definition. As for "interaction technique", Bailly reuses Appert's

definition which, as discussed in the next section, covers half of the phenomena that characterize interaction between a human and a computer system [Appert 04]. Although "parent-menu" is a useful concept to reason about interaction trajectories and menus organization, the notion of root, which is a parent with no parent, is missing from the definitions set. Typically, a menu-bar serves as the root parent for pull-down menus. A menu-bar is not a menu since it is generally not transient. It may thus be interesting to discuss the *compatibility* between a menu and its root parent when this parent is not a menu, but an interactor.

Interactors, interaction objects, widgets, interaction instruments and interaction techniques are now discussed.

6.1.2. Interaction Object, Widget, Interactor, Instrument, Interaction technique

A variety of terms has been introduced since the early days of research in Computer Human Interaction to denote the entities that compose an interactive computer system so that users can accomplish tasks with this system. As for gesture styles (cf. Chapter 2), this "terminology jungle" can be explained by a rapidly evolving field of research, by different research goals, or simply by forgetting or not being aware of early work in the field.

The term "interactive object" or "interaction object" was introduced in the mid-eighties as a software abstraction for the modular implementation of user interfaces [Coutaz 87, Coutaz 91]. Athena "widgets" have been the first concrete implementation of this concept for graphical user interface toolkits based on the object-oriented programming paradigm.

The term "interactor", which was initially introduced by Faconti [Faconti 93] to formalize some aspects of graphics standards, has then be generalized within the European Amodeus project as an abstraction to reason about the behavior of interactive systems from the system as well as from the user perspectives [Coutaz 93]:

From a system viewpoint, *an interactor encapsulates a state which is reflected through a rendering function onto some perceivable representation; it processes input events produced by the environment, and produces output events as responses to the environment* [Duke 93]. Here, the environment denotes the user or other interactors. For example, a mouse is an interactor that encapsulates a position and a button

status. This position is rendered by a cursor on a graphics display. A menu is an interactor that presents a list of options and broadcasts an option when it is chosen. A composed interactor can be built from a mouse and a menu through event bindings. This model has been subsequently related to architectural models for interactive systems, in particular the PAC model [Coutaz 87].

From the perspective of a user, *an interactor is a component in the user interface that mediates between the underlying system and the user. An interactor captures the interaction or dialogue between the two parties, and thus provides a framework for formulating and reasoning about the properties of this interaction* [Duke 95]. This view of an interactor as a mediating entity between a human and the system and its interactive properties is (very) similar to the concept of "interaction instrument" introduced by Beaudouin-Lafon in 2000 [Beaudouin-Lafon 00].

.

Indeed, Beaudouin-Lafon defines *an interaction instrument as a mediator or two-way transducer between the user and domain objects. The user acts on the instrument, which transforms the user's actions into commands affecting relevant target domain objects. Instruments have reactions enabling users to control their actions on the instrument. Instruments also provide feedback as the command is carried out on target objects* [Beaudouin-Lafon 00]. This definition of an instrument is close, in spirit, to that of Cadoz for instrumental gestures (c.f. Chapter 2). An interaction instrument clarifies the distinction between domain-dependent entities, which are of interest in the current task(s), and the tools, which are the entities used to manipulate these objects. By contrast, the interactor model, which is viewed as a unifying concept to reason about system properties, does not make this distinction explicitly.

In addition, an *interaction instrument is composed of a physical part, the input device, and a logical part, the representation of the instrument in software and on the screen* [Beaudouin-Lafon 00]. Again, an interactor models these two aspects in a unified way using composition operators to formally define interactors of any degree of complexity. Mackinlay's *et al.* input device model adopts a similar unifying approach where composed devices, such as a menu, can be modeled as a set of devices related by composition operators: a linear menu is a position linear device that is connected to a cursor virtual device. The cursor is in turn cascaded from the mouse which is a merge of two elementary devices that sense relative movements in x and y plus a layout composition in z for the buttons.

The term "interaction technique" has multiple acceptations, depending on the context of discourse. Foley defines an *interaction technique as a way of using a physical input/output device to perform a generic task in a human-computer dialogue* [Foley 90a]. More recently, Hinckley *et al.* refine Foley's definition by making explicit the binding of input and output hardware devices with software entities: *an interaction technique is the fusion of input and output, consisting of all software and hardware elements, that provides a way for the user to accomplish a task* [Hinckley 04]. According to this definition, a menu is an interaction technique for choosing an item from a set of options using a mouse and a graphics display. In her thesis, Appert proposes a more formal definition: *an interaction technique is a set of interaction steps* where *an interaction step is a sequence of human actions that progressively reduces the command space to a single command* whose execution *leads to a new system state* [Appert 04].

Interestingly, Appert's definition brings forward the incremental reduction of the command space under human control. This progression is not reflected in Foley's nor in Mackinlay's *et al.* definitions. On the other hand, Appert ignores the system feedback which, precisely, helps users to modulate their actions appropriately. As a result, this definition does not capture the mutual influence between human and system actions when engaged in a mutually observable and dependent set of actions that progressively leads to the construction (or abortion) of a command. In addition, Appert's definition does not specify the "depth" of the change of the system state when the command is executed. We can hypothesize that a command matches a task as referenced by Foley and Mackinlay. Implicitly, if we refer to the Arch architectural model [Arc 92], the system state of concern is a change at the Dialogue Control level (which is the technical counterpart of a task model), and possibly at the Functional Core and Functional Core Adaptor levels.

Surprisingly, none of the above definitions address parallel inputs actions as in bimanual interaction. Benko *et al.* propose a bimanual complementary selection interaction technique, the Dual Finger X-Menu [Benko 06]. But the interaction space is still uncovered and few example have been proposed for bimanual parallel interaction techniques.

The subsections above have presented the state of the art about menu definitions and their related concepts. We need now to propose a synthesized view of these definitions while at the same time specifying the choices made for this doctoral research. The intent is to be useful to the clarity of the work, not to be universally accepted.

6.1.3. Synthesized Definitions/Choices

About Interaction objects, Interactors, Instruments, etc.

Interaction objects, interactors, and interaction instruments denote the same fundamental concept but they differ in how they are exploited as a modeling abstraction. Interaction and interactive objects are *architectural units* introduced as an alternative to the sequential monolithic language-based approach to user interfaces implementation. Widgets denote general purpose *reusable building blocks* made available in UI toolkits. Interaction instruments support the analysis of *properties of the interaction*, whereas interactors are intended to reason about both *the internal system properties and the external interactional properties* of the system. Thus, the interactor is a more general unifying concept than the interaction instrument.

On the other hand, the instrument clarifies the distinction between interaction objects that represent domain-dependent entities of interest, and intermediary interaction objects that play the *role of a tool*. This distinction makes it possible to reason about an interactor when used as a tool: How much spatial and temporal indirection does it introduce? How well does its logical part integrate the degrees of freedom provided by its physical part? How compatible are the human physical actions performed on the physical part with the perceivable response of the object of interest? These three properties, degree of indirection, degree of integration, and degree of compatibility are the constituents of an analysis and design framework known as the *Instrumental Interaction Model* [Beaudouin-Lafon 00].

Because menus are generally used as tools to manipulate domain-dependent objects, I will use the concept of interaction instrument. In contrast to Beaudouin-Lafon but in accordance with Lachenal's model of input devices [Lachenal 04], I will consider the physical part of an interaction instrument as a full-fledged instrument and keep explicit the distinction between *physical instruments* and *software instruments*.

Physical Instruments. Physical instruments represent the communication channel between two worlds (the physical human world and the digital metaphorical-based world). Because physical instruments need to live in these two realities, they are composed of two parts: the hardware input device that users can manipulate and its software counterpart that represents the human capabilities in the digital world. A physical instrument is thus a couple <*hardware component, software component*>.

For example, the mouse is an input interaction instrument composed of a plastic real world tangible object that users can manipulate and a software component, the pointer, that represents the human hand capabilities in the digital world. As an interaction instrument, the mouse communicates with the machine, changes its status and controls software instruments. In its more complex versions, the mouse is enriched with several buttons or scroll-wheels each of them constituting a well-defined physical instrument. For post-WIMP interfaces, touch screens are the hardware component while the logical component are the points they generate. The logical part does not necessarily have/need a graphical representation because of the direct absolute pointing function of touch screens.

Modeling input devices as a couple <*hardware component, software component*> has been used before. The Amodeus elementary interactor presented above conveys a similar model. The implementation of Dragicevic's Icon Toolkit [Dragicevic 04] is based on the same separation of concerns.

Software Instruments. Software instruments live in the logical (software) world only, and are usually inspired from real world instruments. In a classical WIMP environment, software instruments are driven by keyboard and mouse physical instruments. Scroll-bars, software buttons, menus, etc. are examples of software instruments driven by the logical components of physical instruments. In modern post-WIMP environments, software instruments are driven, among others, by touch screens and accelerometers based physical instruments.

For the purpose of this doctoral research, i.e. gestural based interaction combined with multi-point screen handheld devices, the reference model I will use is a two level composition of a Physical Instrument with a Software Instrument. People interact with Physical Instruments through their hardware component. The software component of physical instruments represents the extension of human capabilities in the digital world. Through the use of such physical instruments users act on software instruments to accomplish their tasks.

We need now to define a menu as a software instrument. Given that this doctoral research focuses on graphics output devices, vocally rendered menus are not considered.

About Graphical Menus

The following revised version of Bailly's definition is proposed: A graphical menu is a spatially-activated software instrument, without any temporal side-effects, that provides users with access to the complete set of possible software instruments they can use on the domain objects the menu refers to (a subset of the user contexts). Where:

software instrument : a software instrument designed to be connected to a physical instrument;

spatially-activated : because its activation needs to happen in well-defined areas of the screen;

without temporal side-effects : by contrast to palettes, menus do not enter operational modes temporally invariant until an explicit change (even if, at a lower level of description, the item selection process can be temporally driven);

proposing ... the complete library of software instruments : with respect to other software instruments, a menu proposes an exhaustive set of software instruments;

domain objects : the set of potential objects of interest for the user;

the menu refers to : while contextual menus contain the exhaustive set of software instruments that can be activated in the context they refer to, system menus embrace a wider domain thus refer to multiple contexts.

In addition, a distinction must be made between the menu selection process and the menu *per se*. They are two distinct bricks of the same software instrument. The menu selection process is the interaction technique implemented to support the *selection* of a menu item. The menu is the software (widget) implemented to offer the selectable choices: it is the multiplexer of the selection interaction technique.

In synthesis:

A gesture is a physical instrument whose hardware component is a direct input device (mouse, pen or touch screen) and the software component, a gesture recognizer.

A menu is a software instrument that satisfies the following requirements:

1. It supports the selection of one item within a finite set of options.

2. It minimizes mental efforts by presenting the set of options to the user (favors recognition as opposed to recall).
3. It presents the options as a semantically and spatially meaningful structure that is supposed to match human goals.
4. It is transient in the sense that it is perceivable to the user only during its interaction with the user.
5. It is quasimodal since it defines a local context for system interpretation that is maintained explicitly by the user until the interaction ends.

.

A gestural menu is a menu software instrument connected to a gesture instrument such that the selection process is driven by gestures.

In the remaining of this chapter, I analyze the classification spaces provided for menus in general followed by gestural menus.

6.2. Two taxonomies for Menus

Two taxonomies are considered for their complementarity: that of Shneiderman who focuses on menus structure, and Bailly's MenUA who adopts an instrumental approach to the analysis of the menu design space.

6.2.1. Shneiderman

In his description of *Menu selection systems*, Shneiderman stresses *the importance of a meaningful organization for menu items*. In Shneiderman's words, *the primary goal for menu designers is to create a sensible comprehensible, memorable, and convenient semantic organization for the user's tasks*. Shneiderman has identified five possible *semantic organizations* for menus (see Figure 6.2):

Single Menus have two or more items that represent the final command, action or option available in the current context.

Linear Sequence of menus is basically represented by a form. Different options need to be chosen in order to specify the arguments needed by the chosen command.

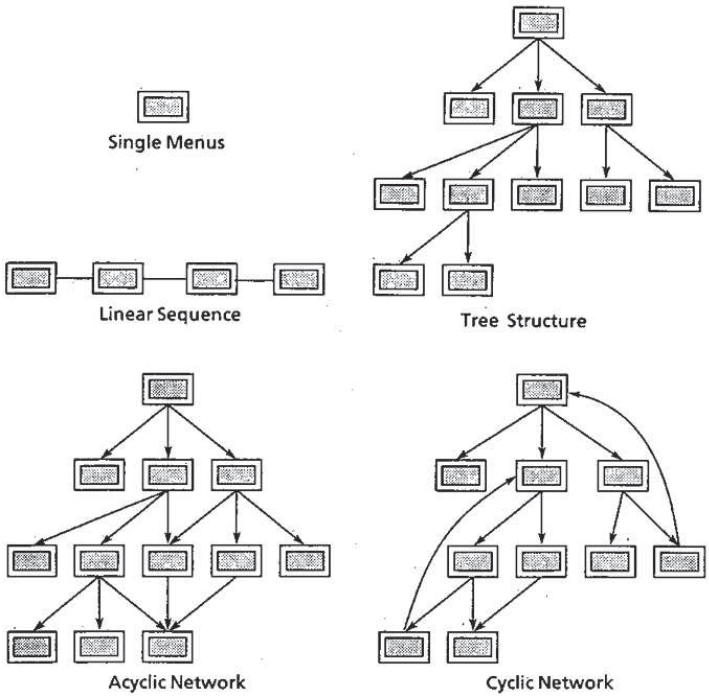


Figure 6.2.: Different menu structures impose different approaches and learning difficulties on users [Shneiderman 87].

Tree Structure *When a collection of items grows and becomes difficult to maintain under intellectual control, people form categories of similar items creating a tree structure.* Although trees seem to be naturally acceptable for human brain, it is not simple to choose the right grouping. Some people could be unfamiliar with the chosen organization thus having different difficulties while approaching the menu.

Acyclic and Cyclic Network structures propose the most difficult approach to menus organization. Users can easily get lost as they do not completely understand the structure, thus feeling frustrated.

Shneiderman refines the Single Menu class along two lines of analysis: (1) the number of items and (2) the appearance of these items. Single menus can be sub-classified according to these axes:

Binary Menus propose a two option choice; they usually prompt the user, offering either a feature (*DO YOU WANT INSTRUCTIONS?*), or a parameter choice (*Choose ordering index: N: NAME, D: DATE*). Figure 6.4 is an example of a binary menu displayed through a WIMP pop-up alert panel.

Extended Menus propose the most frequent options to the users, then the less frequent ones in a subsequent distinct interaction space.

Pop-up or Pull-down menus *appear on the screen in response to a click performed with a pointing device* [Shneiderman 87].

Permanent Menus occupy a fixed space, providing users with the most used commands or options. By contrast with extended menus, they are not the continuation of a menu, but they form their own menu depending on the current context. Permanent menus are best known as palettes.

Multiple Selection Menus let the user choose different options simultaneously.

Considerations are summarized about tree structured menus as well. Four principles for designing the hierarchical organization of tree menus are proposed:

Depth versus breadth. Based on the analysis of several performance studies, Shneiderman concludes that *fewer* [depth] *level aid decision making*, without ignoring the semantic structure of the items.

Semantic grouping in trees In order to best fit users expectations, designers need to consider the following empirically validated rules:

1. Create groups of logically similar items;
2. Form groups that cover all possibilities;
3. Make sure that items are non-overlapping;
4. Use terminology familiar for the target users, but make sure that the items are distinctive from each other.

Menu maps. *As the tree structure grows, users have greater difficulty in maintaining an overall understanding of the semantic organization ... Offering a spatial map can help overcome this difficulty* [Shneiderman 87].

Semantic versus alphabetic organization. Informed by empirical evaluations of human performance, Shneiderman suggests the use of semantic organization for a first level grouping while the alphabetical order is best suited for the second level organization. As shown in Figure 6.3, modern hierarchical menus groups items semantically.

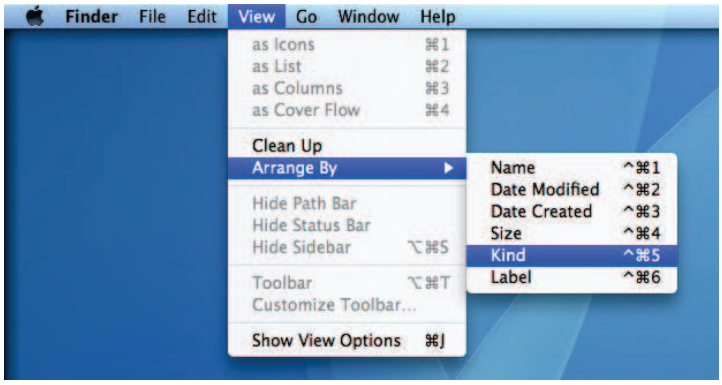


Figure 6.3.: Shneiderman's hierarchical menus in a modern WIMP environment.

1	Use task semantics to organize menu structure (single, linear sequence, tree structure, acyclic networks, and cyclic networks)
2	Try to give position in organization by graphic design, numbering and titles
3	Items become titles in walking down a tree
4	Make meaningful groupings of items in a menu
5	Make meaningful sequences of items in a menu
6	Items should be brief and consistent in grammatical style
7	Permit type-ahead, jump-ahead, or other short-cuts
8	Permit jumps to previous and main menu
9	Use consistent layout and terminology
10	Consider novel selection mechanisms and devices
11	Consider response time and display rate impact
12	Consider screen size
13	Offer help facilities

Table 6.1.: Shneiderman's menu selection guidelines distilled from practice [Shneiderman 87].

Shneiderman also considers the variation of items presentation, as well as different ways of increasing direct access to menu items, including consideration for response time, display rate and shortcuts.

Although Shneiderman's classification is somewhat influenced by the technology of the eighties, the recommendations listed in Table 6.1 still serve as a reference for **Menu Selection Guidelines**.

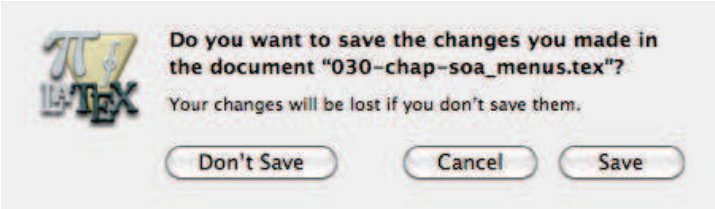
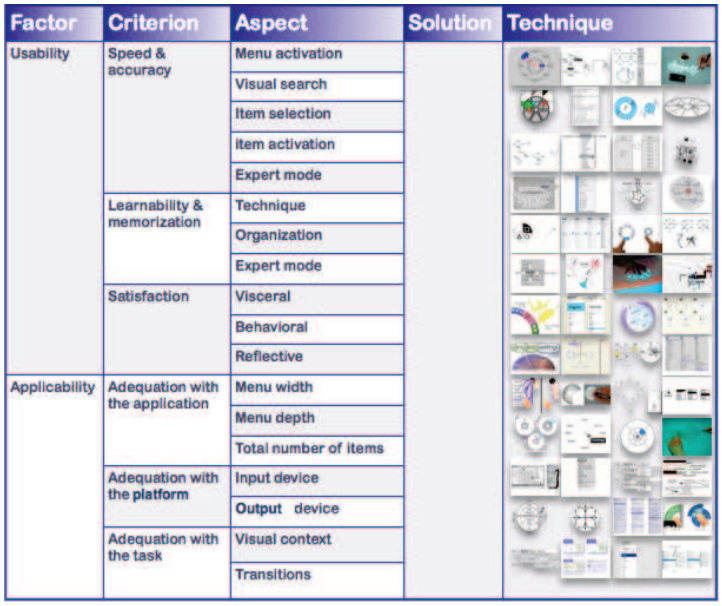


Figure 6.4.: Shneiderman's binary menus in a moder WIMP environment.



Factor	Criterion	Aspect	Solution	Technique
Usability	Speed & accuracy	Menu activation		
		Visual search		
		Item selection		
		Item activation		
		Expert mode		
	Learnability & memorization	Technique		
		Organization		
		Expert mode		
	Satisfaction	Visceral		
		Behavioral		
		Reflective		
Applicability	Adequation with the application	Menu width		
		Menu depth		
		Total number of items		
	Adequation with the platform	Input device		
		Output device		
	Adequation with the task	Visual context		
		Transitions		

Figure 6.5.: Bailly's MenUA classification space [Bailly 09].

From Shneiderman's early work in the area of menu design, two contributions must be kept in mind: 1) The classification of menus along two main axes: structure and visual appearance. 2) The importance of a coherent presentation so that users understand and foresee the menu behavior.

6.2.2. Bailly's MenUA design space

Building on the instrumental interaction model, Bailly [Bailly 09] proposes MenUA (Figure 6.5), a classification space that characterizes menus from their external features rather than based on their internal structures or feedback characteristics. Bailly approaches menus according to their capacities to adapt to two classes of constraints: constraints from the target users, and constraints from the system side. As a consequence, the first level organization of MenUA is composed of two *Factors*: *Usability* and *Applicability* (thus, the name MenUA where U stands for Usability, and A for Applicability). The

Usability factor encapsulates the features that characterize the adaptation capabilities of menus systems against the cognitive, motor and sensory abilities of the target user[1]. By contrast, the *Applicability* factor (often called utility in the literature) groups menu features that determine the adaptation capabilities of the menu system to the user's functional needs[2].

Usability and Applicability both cover three criteria. In particular, Usability is refined into the following criteria:

Speed & Accuracy measure the effectiveness of the menu in supporting the user to select commands[3].

Learnability & Memorization represent the capacity of the menu to allow the user to make optimal use of it rapidly and sustainably[4].

Satisfaction represents the ability of the menu to provide pleasurable feeling that results from task closure[5].

In turn, Applicability covers the following three criteria:

Adaptation to the application describes the capacity of the menu to contain the commands of the underlying application[6].

Adaptation to the platform describes the capacity of the menu to be used with different input/output devices[7].

Adaptation to the task describes the capacity of the menu to adequately fit within the user's task space[8].

Finally, criteria are further refined into more practical features called *Aspects* that can be measured experimentally. For example, Speed and Accuracy are measurable in terms of menu activation (or invocation), visual search, item selection (which characterizes the interaction trajectory and the time to reach the appropriate item), item activation (which triggers the execution of the system command), and the existence or absence of

[1] Adéquation du cystème par rapport aux capacité cognitives, mortices et sensorielles de l'utilisateur [Bailly 09].
[2] Adéquation du cystème par rapport aux besoin applicatifs de l'utilisateur [Bailly 09].
[3] Efficacité du menu pour permettre à l'utilisateur de sélectionner des commandes [Bailly 09].
[4] Capacité du menu à fournir un sentiment agréable qui result de l'accomplissement de ce sue l'on souhaite [Bailly 09].
[5] [Bailly 09].
[6] Capacité du menu à contenir les commanfes de l'application [Bailly 09].
[7] Capacité du menu à fonctionner aver différents dispositifs d'entrée et de sortie [Bailly 09].
[8] Capacité du menu à s'intégrer effacement dans la tâche de l'utilisateur [Bailly 09].

an expert mode. As shown in Figure 6.5, MenUA proposes eighteen Aspects capable to structure the whole state of the art related to graphical menus.

Using MenUA, designers and developers are able to choose the most appropriate menu(s) by simply analyzing the requirements (target users and system constraints) imposed by their project, thus identifying the key relevant *Aspects*. For example, a technical/interactional solution that improves the "item selection" aspect, improves the overall usability of the menu according to the speed & accuracy criterion. A menu with a large "menu depth" and a large "menu width" improves its overall applicability.

The leaves of MenUA classification abstract away Shneiderman's guidelines. For example, Shneiderman's hints for semantic grouping of items in menu trees is just one solution to the MenUA *Visual Search* aspect, promoting a better user experience in the usability of the menu itself.

Shneiderman's versus Bailly's design and classification criteria mirror the evolution of the concept of menu across the years as well as that of user interfaces. Shneiderman addresses all kinds of interactors that support selection tasks. By contrast, Bailly narrows down the analysis to a particular subclass of interactors referring to commands/functionalities rather than to arguments. In modern user interfaces, cyclic and acyclic menu structures are rarely used. Binary menus found their concrete implementation in alert panels, while linear menus correspond to form panels. Single menus are just a special case of hierarchical menus which are at the center of Bailly's research.

In synthesis, the key point to retain from these two classification spaces is the concern for human-centered issues as elements for charactering menus: Users behavior facing a new widget (*Satisfaction*), human cognitive capabilities compared to menu organization and structure (*Learnability & Memorization*), as well as detailed characteristics for the selection process (*Speed & Accuracy*). The Aspects that derive from the Usability factor fully characterize the parameters of the interaction technique offered by a menu.

6.3. Gestural Menus

Menus and gestures analysis find their intersection in Gestural Menus. Gestural menus share the same organization, properties and problems than conventional menus and they use gesture physical instruments for the selection process. Intrinsics properties of gestural organization and recognition need to be considered in order to obtain seamless integration

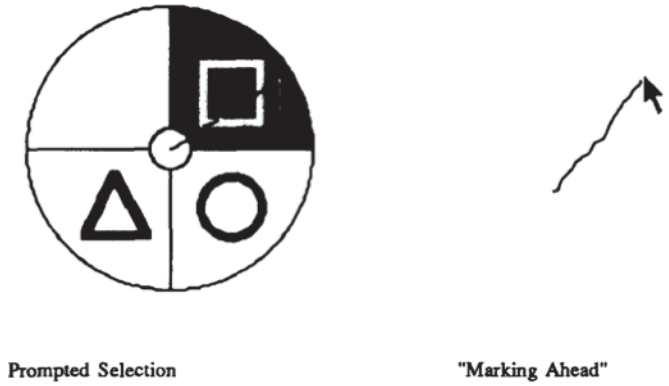


Figure 6.6.: Marking menus propose a Novice and an Expert mode. On the left, a marking menu used in novice mode resembles a Pie Menu [Callahan 88] but user's movements are tracked leaving an ink trail. On the right, the marking menu when used in expert mode: the user makes a mark without relying on the graphical representation [Kurtenbach 93].

for enhanced user experience. The following subsections refer to five representative gestural menus studies in order to understand and structure the argument. In my opinion, Kurtenbach's Marking Menus, Bau's Octopocus, Appert's gestural shortcuts, Bailly's Flower and Leaf Menus and Roudaut's synthesis represent a complete framework that introduces the problems, their evolution and a synthetic state of the art on gestural menus.

6.3.1. Kurtenbach's Marking Menus

Marking menus are considered as the first example of successful gestural menus. With the main objective of reducing the gap between novice and expert users, Marking menus share the structure of pie menus but instead of focusing on the current position relatively to the center, they make expert users focus on the gesture they have to perform to select an option. In Kurtenbach's words: *Marking menus are invisible pie menus in which the movement of the cursor during a selection leaves an "ink trail" similar to a pen stroke on paper* [Kurtenbach 93].

Physical and cognitive issues arise when the user needs to face invisible menus. First, they need to remember the structure of the menu as well as the relative position of each item. Second, they need to associate marks with the command they represent: *Subjects are faced with the task of either mentally representing the menu or associating marks with the commands they invoke through practice.*

As a result, gestural menus need to be "self-revealing" in order to let users discover and learn them. *Menus and buttons, for example, are self-revealing: the set of commands is readily visible as a by-product of the way commands are invoked.* By using pie menus as a revealing mechanism for marking menus, Kurtenbach permits novice users to learn while freeing expert users from being annoyed by a graphical representation that pops over the working context.

From Kurtenbach's work, the take away message is threefold: design gestural menus that support self-revelation, guidance, and rehearsal. In addition, as discussed in Chapter 3, even simple gestures can express complex commands. Kurtenbach observes that the start and end points as well as the length/speed of the stroke *can be used to communicate additional information* including command parameters.

The original marking menu has inspired many extensions such as the "donut menu" suggested by the same authors [Kurtenbach 93] and more recently, the wavelet menu using concentric circles rather than menu items to browse through hierarchical levels [Francone 09].

Bau addresses *the problem of remembering which mark does what* through feedforward while Appert displays a mark that represents the gesture next to menu items, just like accelerator keys.

6.3.2. Bau's Octopocus

OctoPocus is a gestural menu that combines feedforward and feedback in a tightly coupled continuous manner to help users to learn, execute and remember gesture sets Figure 6.7. The motivation is to bring users from novice to expert level performance in a graceful and efficient manner [Bau 08]. In contrast to feedback whose purpose is to represent the current system state, feedforward represents the possible states the system can be in the future depending on the next user's actions.

Bau uses feedforward combined with feedback as a way to support a dynamic form of self-revelation. In [Bau 08], he proposes a taxonomic space for characterizing feedforward and feedback in relation to gestural interaction. For example, feedforward approaches can be plotted in a two dimension space: first, the level of detail provided to the user (direction only as for the Marking menus, portion of gesture as for OctoPocus, full gesture description as Appert's StrokeShortcuts described in the next section); second, the update rate (only once for marking menus, in multiple steps as in hierarchical marking

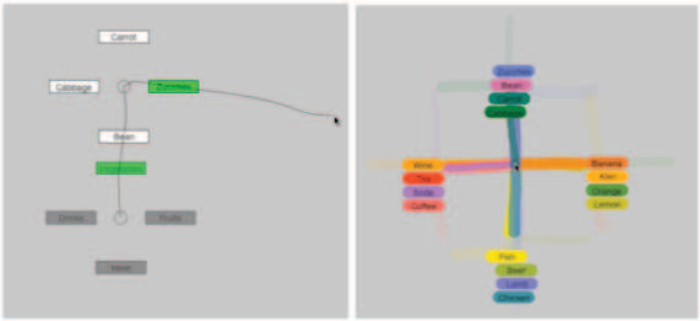


Figure 6.7.: A hierarchical marking menus (left) enriched with feedforward (right) that permits the user to foresee each command's hierarchical content [Bau 08].

menus, and continuously for OctoPocus). The taxonomic space for feedback addresses the different phases of gesture recognition with different forms of representation including mark beautification, recognized paths, and labels.

Bau's taxonomic space displays the lack of well-designed feedback and feedforward mechanisms for existing gestural interaction techniques. In particular, the feedback provided by Hierarchical Marking menus reveals that there is a lack of continuous communication between the system and the user, thus generating a simple *binary recognition value*: they *update ... low-details hints in discrete steps as the user progresses through the menu hierarchy*. Existing solutions either do not provide continuous feedback/feedforward in response to user's actions, or they provide verbose feedback/feedforward information that occludes the user's context, thus disturbing the interaction itself.

Bau's solution consists in proposing *dynamic guides ... providing ... feedforward about the user's current set of options and feedback about how well the current gesture has been recognized*. An example of a dynamic guide applied to a hierarchical marking menu is shown in Figure 6.7. The figure shows the first hierarchy level of the menu as the user has just started the interaction. Semitransparent marks that show the content of the next level hierarchy, serve as a feedforward guideline.

From Bau's dynamic guide principle, the take away message is the importance of well-designed feedforward that is continuously and tightly coupled with feedback to help users learn, discover and remember gestural menus.

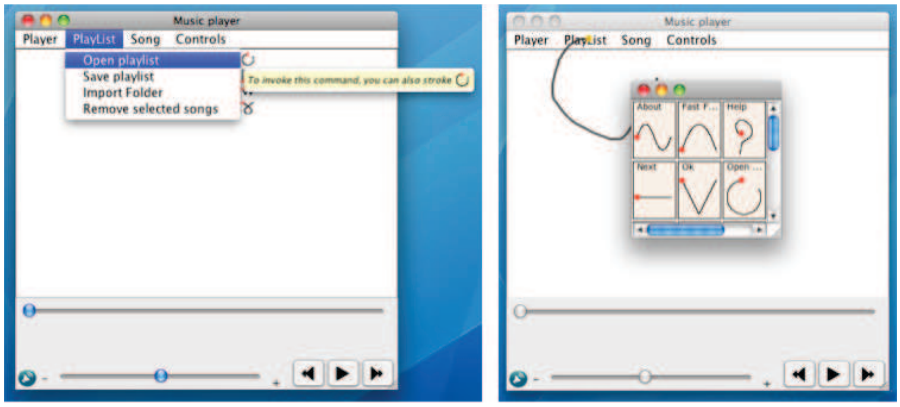


Figure 6.8.: StrokeShortcuts integrated in a media player. On the left, gesture shortcuts are visible while using a classical linear menu. On the right, the "open playlist" corresponding gestures. Help fades away on a timeout.

6.3.3. Appert's gestural shortcuts

Classical WIMP menus apply the "self revealing" principle by showing the keyboard shortcuts conjointly with their corresponding item labels. What if the expert mode of interaction is gestural? Kurtenbach has proposed to improve gestures learning by popping up *a menu that lists commands with their corresponding marks* [Kurtenbach 93]. Similarly, Appert proposes to augment linear menus with their corresponding stroke representations [Appert 09] (see Figure 6.8). She demonstrates that stroke shortcuts are easier to memorize than keyboard shortcuts. As important, she proposes *guidelines to implement gestural shortcuts easily.*

Template-Based recognition algorithm. By eliminating training issues while still being accurate, a template-based algorithm is the best choice to implement stroke shortcuts;

Simplify the task of designing a set of strokes. By offering designers a gestural implementation design space, their imagination is stimulated while increasing the robustness of the recognizer (i.e. the design space avoids the definition of gestures that are too similar);

The underlying mechanisms in the recognition engine must be transparent to the interface designers. This recommendation complements the existence of a design space, above;

Make stroke shortcuts visible to end users. If gesture based GUIs follow feedforward principles, gestural vocabulary is better assimilated by end users;

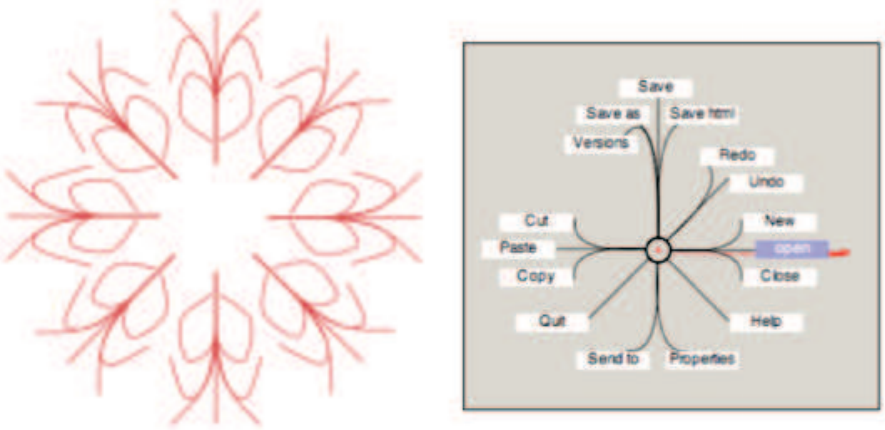


Figure 6.9.: Ideal Flower menus support up to 56 items per hierarchy level (left). Flower menus can control the most used commands in a simple hierarchy level (right) [Bailly 09].

Integrate stroke shortcuts in graphical toolkits. If gestural shortcuts are integrated into existing graphical toolkits at the appropriate level of abstraction, then developers can provide them in their user interfaces at no additional development cost.

From Appert's work, the take away message is her guidelines that provide a practical answer to the problem of making gestural interaction available to end users and to programmers, complementing Kurtenbach's "self-revealing" principle. The integration of gestural interaction into existing toolkits is a fundamental requirement for involving designers in the experimentation of gestural interactions as well as for integrating gestural interaction seamlessly into existing environments. In Chapter 7, I will show how GeLATI has been designed to integrate gestural interaction in native UIKit based widgets.

Appert's approach finds its concretization in another representative example of gestural menus: the Leaf menus.

6.3.4. Bailly's Flower and Leaf Menus

Among the properties of gestures that have already been studied and implemented (e.g., length, direction, and speed), Bailly exploits an original characteristics: curvature which has a discriminating property of the stroke together with the angle and direction.

As shown in Figure 6.9, flower menus improve marking menus in terms of the number of items that each hierarchy level can support. The proposed interaction is compatible with the classical interaction technique based on object clicking.

Flower menus share with marking menus the interaction principles: when the user starts the interaction the menus is displayed at the location specified by the user. In order to support the expert mode, the graphical representation is shown after a time delay (100ms[9]). As an evolution of marking menus, a flower menu can contain up to 56 items and organizes them in different internal groups (four main groups). A hierarchical version of the flower menu is proposed using a concentric mechanism similar to the that of the Donut Menus.

Using MenUA as a framework, flower menus support the Usability factor in the following way: (1) They enhance visual search thanks to the internal grouping; (2) From marking menus, they inherit the simple item selection; (3) They support learning and memorization as they engage users into a simple interaction technique and let them foresee the gesture shape through the menu structure itself. From the applicability perspective: (1) they can easily support the application functional needs since they can contain up to 56 items per hierarchy level and more than one level of hierarchy; (2) they are adequate for multiple platforms since they require a 2D pointing physical instrument; (3) they are adequate for the task by proposing different command activation systems and visual representations.

However, the visual footprint of the Flower menu may be an impediment for small screen mobile devices. As an answer to this limitation, Bailly and Roudaut propose an evolution of marking menus: the Leaf Menu that brings together the gestural concepts of the flower menus and Appert's guidelines.

The Leaf menu is controlled by gesture strokes. Gestures are characterized by direction and curvature. Items are grouped and gesture shortcuts follow simple rules thus simplifying learning and memorization. Gestures are compatible with the use of conventional linear menus, which guarantees easy access to the novice user. Gestural shortcuts are visible to users in novice mode, which supports smooth novice-to-expert transitions. Leaf menus can contain up to seven items in one hierarchy level. As shown in Figure 6.10, they are able to adapt the graphical representation as well as the gestural shortcuts, according to the nearest screen edges and activation point.

From the Leaf menu, the take away message is the use of curvature as a means to discriminate between gestures and the adaptation of gestural menus to mobile platforms.

[9]Bailly [Bailly 09] uses this value in his Flower menu. Although it seems to be a valid and widely used approximation it still need a scientific experimental validation

Figure 6.10.: Leaf menu for touch-screen enabled mobile devices where curvature is exploited to discriminate gestures [Bailly 09].

We close this section with Roudaut's contribution to the analysis of the state of art about gestural menus.

6.3.5. Roudaut's Analysis

As shown in Table 6.2, Roudaut classifies gestural menus within a two dimension space: the gesture style (which may be semaphoric or deictic, cf. Chapter 2, Table 2.7), and the graphical layout used for rendering the structure of the menu (which may be linear, circular, or semicircular).

Conventional linear menus, which are position driven, use pointing to express the choice of the desired menu option. As a result, the gesture associated to a linear menu is deictic. They are organized according to a linear structure where each command is positioned at a variable distance from the activation point to reflect their index. ThumbMenu and ArchMenu are also classified as deictic and position driven. Nevertheless, they are characterized by a semi-circular rendering structure as their options are laid out along an arc at the lower left (right) part of the screen. Such an organization permits to have all menu items at the same distance from the activation point independently of their

	Deictic	Semaphoric
Linear	Linear Menu	Linear Menu with Gestural Shortcuts LeafMenu
Circular		Marking Menus Compound Marking Menus Multistroke Menus Wavelet QuickWriting Flower Menu PushMenu TiltMenu
Semi-Circular	ThumbMenu ArchMenu	

Table 6.2.: Roudaut classifies gestural menus according the gestures type they are driven by and the layout menu items are disposed on [Roudaut 10].

indexes. Furthermore, the activation finger does not occlude any important contextual information.

Semaphoric gestures occupy most of the table. The Linear Menu augmented with Gestural Shortcuts and the LeafMenu are characterized as semaphoric. As discussed in previous subsection, the augmented Linear menu uses a diversified gestural vocabulary in order to provide linear menus on mobile devices with expert mode activations. The Leaf menu uses the same concepts but proposes a uniform gestural vocabulary. Indeed, items can be accessed by stroking curves using different curvatures according to the index of the desired item inside the menu. Circular semaphoric menus include the Marking menus, the Compound marking menus and the Multistroke menus. They share the same menu structure but use different activation and selection interaction techniques and propose different hierarchy access solutions. Among others, we also note the presence of the Flower menu that can be viewed as an extension of the LeafMenu around four axes. The Wavelet menu falls in the same semaphoric circular category as Roudaut classifies it as a circular menu that optimizes the Multistroke menu[10] . As for QuickWriting menu, PushMenu and TiltMenu, their originality comes from the use of other characteristics

[10]Despite the global interaction of the Wavelet menu resemblance with the Multistroke menu. We consider it as a position based menu as ThumbMenu and ArchMenu. As such, we are more tempted to plot it as a deictic menu with a circular item structure.

	Pros	Contras
Linear	Well known technique Space optimized Contextual or System wide Never out of screen Expert and Novice Modes	Lack of precision Limited by small screen size Finger Occlusion Context Occlusion
Circular	Coherent Expert mode (Gestural)	Can appear out of the screen Space management Limited number of items Lack of localized counterpart Finger Occlusion Context Occlusion Articulation Difficulties in some movements
Semi-Circular	Avoid finger occlusion	Lack of expert mode Can appear out of the screen Limited number of items Lack of contextual counterpart Articulation Difficulties in some movements

Table 6.3.: The different layouts gestural menus are characterized by, imply consequences on the interaction they propose. Here a synthesis of the pros and cons proposed by Roudaut [Roudaut 10].

(respectively, the distance, the stylus pressure and the stylus slope), rather than the simple stroke, to define the desired item.

The design choice between deictic and semaphoric styles is driven by the implementation algorithm and by the desired interaction model. With regard to graphical rendering, the choice is not as simple and straightforward. As shown in Table 6.3, Roudaut synthesizes the pros and cons for each layout option.

Linear menus offer the same interaction technique people have learnt from their desktop computer experience. Their footprint on the screen is more compact than that of the circular menus as it occupies half of the space taken by a circular menu. Linear menus offer a contextual mode, activated near the objects of interest, as well as a system wide mode, usually available through a menu bar placed near the vertical edges of the display. Both versions are usually designed not to exceed the display real estate (when a linear contextual menu is opened in a non optimal position, it is re-centered in order to be completely visible). Moreover linear menus propose graphical interface independent expert mode (as we have seen the keyboard shortcuts can be substituted with gestural ones) allowing users to memorize and access the desired options gradually and directly.

On touch-screen enabled mobile devices different characteristics need to be taken into account. Small screen, lack of keyboard for menu accelerators, absolute pointing devices, hand and finger occlusion are just few of them. From Roudaut's work, we retain the classification of menu structures contextualized for mobile environment.

6.4. Synthesis

In this chapter, I have introduced the problems related to gestural menus by analyzing the interaction techniques involved in the menu selection process, by presenting the different points of view from which menus have been analyzed, and by considering some of the key taxonomies developed to characterize them.

Given the large number of definitions for the concept of menus and for its parent class, the first contribution of this chapter is to propose a consistent set of definitions based on representative works in the domain. I propose to define *a menu as a software instrument connected to a physical instrument* where software and physical instruments are interaction instruments as introduced by Beaudouin-Lafon. Because physical and software instruments have very distinct properties, I consider it important to make a clear distinction between those objects that bridge the gap between the real world and the digital world from those that live in the digital world only.

Therefore, a menu is a digital "beast" and *a gesture is a physical instrument whose hardware component is a direct input device and the software component, a gesture recognizer*. In addition, *a menu software instrument satisfies the requirements as specified by Bailly (i.e. supporting the selection of one item within a finite set of options, presenting*

these options in a semantically and spatially meaningful structure, being transient and quasimodal). As a result, *a gestural menu is a menu software instrument connected to a gesture instrument such that the selection process is driven by gestures.*

Having provided definitions for menus, gesture, and gestural menus, the second contribution of this chapter is to demonstrate the complexity of menus by the way of two complementary taxonomies: that of Shneiderman who discusses various forms of menu structures and Bailly's MenUA that characterizes menus according to their appropriateness to human capabilities and to human functional needs.

The third section of this chapter presents a brief analysis of the state of the art for gestural menus ranging from Kurtenbach's seminal work to Roudaut's synthesis of gestural menus for mobile devices. From the interaction point of view, the take-away message is to *design gestural menus that are self-revealing, that support dynamic guidance and rehearsal, as well as smooth migration between novice and expert levels of expertise.* From the implementation point of view, one should consider the use of a template-based approach to gesture recognition in order to avoid system training, the use of curvature for discriminating gestures, as well as the problem of integrating gesture in conventional GUI toolkits so that developers can provide them in their user interfaces at no additional cost.

To conclude, I propose to structure the lessons and recommendations from earlier work into the following three axis framework: physical layout of menu items, graphical rendering, and interaction style. These axes respectively called **Structure**, **Aspect** and **Interaction** have been used to organize GeLATI. GeLATI is a template driven gestural library to design, prototype and implement crossmodal and multimodal gestural menus. This contribution is presented in detail in the next chapter.

Chapter 7.

GeLATI: integrating hierarchical gestural menus in existing toolkits

In this chapter, I present GeLATI, a Gestural Library for implementing gestural interaction based on accelerometers and touch screen devices[1]. GeLATI can be characterized in the following way:

1. Gesture recognition is based on a new real-time vectorial approach that combines multiple inputs to support cross-modal and/or multi-modal interactions;
2. Gesture recognition, which uses a single-sample template approach, does not need training;
3. Gestures, which are modeled as series of vectors, are rectilinear. As demonstrated by Bragdon *et al.* [Bragdon 11] (c.f. Chapter 2), rectilinear gestures are more usable than free form gestures;
4. Both deictic and semaphoric gestures are supported;
5. Novice users are guided along well-defined paths to complete a gesture stroke. Expert users can stroke commands without graphical feedback as exemplified by the Marking Menus;
6. Exploration is supported through an incremental back-to-hierarchy-node mechanism;
7. Hierarchical gestural menus can be automatically integrated into existing graphical toolkits.

[1]GeLATI: Gestural = Ge, Library= L, Accelerometer= A, Touch=T, Interaction=I.

Most existing gestural interaction toolkits rely on statistical recognizers. This approach has been validated through many years and applied to several prototypes. However, from the interaction perspectives, it comes with several limitations. First of all, it requires users to train the system. Second, it is primarily based on post-analysis of users actions ("off-line, batch processing" recognizers). As a consequence, it cannot support tightly coupled interaction as required by the dynamic guidance and reversibility principles. In particular, feedback-feedforward cannot be provided, and users cannot go back to correct their strokes and to explore the gestures set.

In this chapter, I will examine these characteristics and their importance for gestural menus. I will analyze typical fundamental gestural recognizers and show how GeLATI brings in new improvements.

7.1. Existing approaches

This analysis of gestural recognizers is illustrated by two examples from research: Rubine's feature-based recognizer [Rubine 91] and the $1 low-cost recognizer [Wobbrock 07]. Rubine's recognizer serves as a reference that has inspired the development of several gestural recognizers both in academics and industry. The $1 recognizer marks the recent evolution of features driven gesture recognizers, optimized for different devices and available on multiple platforms.

7.1.1. Rubine's GRANMA

GRANDMA is one of the very first efforts in integrating gestural interaction into graphical toolkits to enhance direct manipulation WIMP user interfaces. Rubine's algorithm uses *classification attributes* (such as orientation, size, speed) and event types (such as mouse button up, timeout) to recognize single stroke gestures produced with a direct pointing devices such as a mouse. The gesture designer that comes with GRANDMA allows developers to prototype and test gestural interactions through examples. *Empirical evidence suggests that 15 training examples per gesture class is adequate* [Rubine 91]. Rubine's gestures recognizer uses statistical analysis to discriminate among the gesture set. Gestures are modeled as a set of 13 empirically defined features (see Figure 7.1):

(f_1) The cosine of the initial angle of the gesture;

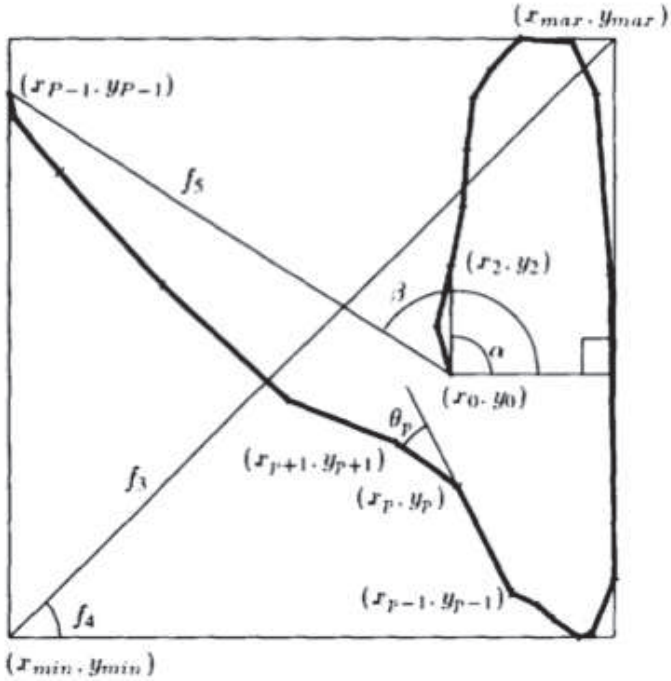


Figure 7.1.: A set of 13 features characterizing each gesture in GRANDMA statistical gestures recognizer [Rubine 91].

(f_2) The sine of the initial angle of the gesture;

(f_3) The length of the bounding box diagonal;

(f_4) The angle of the bounding box diagonal;

(f_5) The distance between the first and the last point;

(f_6) The cosine of the angle between the first and the last point;

(f_7) The sine of the angle between the first and the last point;

(f_8) The total gesture length;

(f_9) The total angle traversed;

(f_{10}) The sum of the absolute value of the angle at each mouse point;

(f_{11}) The sum of the squared value of those angles;

(f_{12}) The maximum speed (squared) of the gesture;

(f_{13}) The duration of the gesture.

Several variations of the original recognition algorithm have been developed. In particular, an "eager" version of the algorithm proposes a gesture as soon as it is unambiguous. In this case, classification is performed on every mouse point event. Another version is a multi-finger implementation where multi-finger input is modeled as multi-path data. *The single-stroke recognition algorithm is thus applied to each path individually while the results are combined to classify the multi-path gesture* [Rubine 91].

More recently, efforts in optimizing single strokes gesture recognizers have led to Wobbrock's $1 recognizer.

7.1.2. Wobbrock's $1

Wobbrock *et al.* has defined eight key requirements that gesture recognizers should satisfy:

1. *be resilient to variations in sampling due to movement speed or sensing;*
2. *support optional and configurable rotation, scale, and position invariance;*
3. *require no advanced mathematical techniques;*
4. *be easily written in few lines of code;*
5. *be fast enough for interactive purposes (no lag);*
6. *allow developers and application end-users to "teach" it new gestures with only one example;*
7. *return an N-best list with sensible scores;*
8. *provide recognition rates that are competitive with more complex algorithms.*

To satisfy these guidelines, the $1 recognizer is structured as a *four step process*:

Resample the Point Path : this step transforms each gesture point path (and template) sampled by the input device into point paths defined by a fixed amount of points N. This step eliminates sampling differences due to variations in speed and/or input device characteristics (e.g., resolution);

Rotate once based on the "Indicative Angle" : the indicative angle is defined as *the angle formed between the centroid of the gesture* $(\overline{x},\overline{y})$ *and the gesture's first point.*

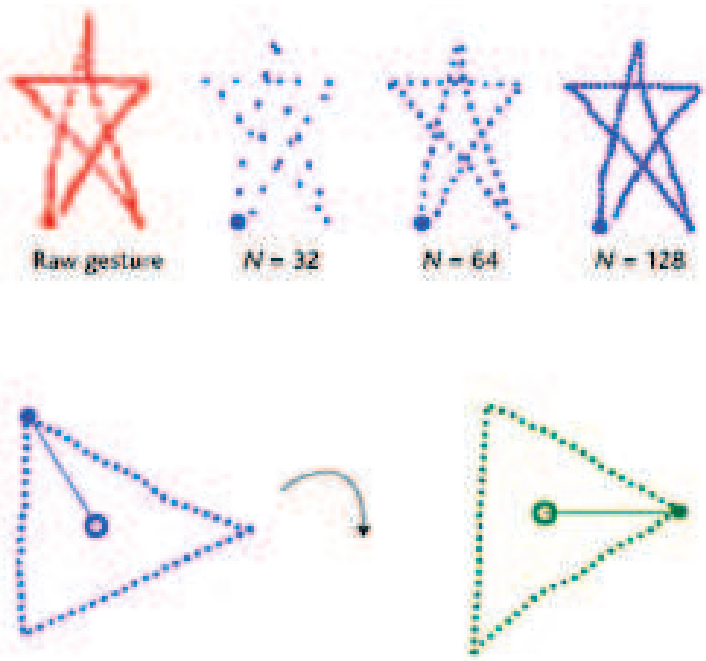


Figure 7.2.: $1 gesture recognizer first two steps. At the top, a raw gesture as captured by the input device with three different resamples where N denotes the number of sampling points. Wobbrock *et al.* have determined empirically to use $N = 64$ as a reference resampling parameter. At the bottom of the figure, the second step of the gesture recognizer. The resampled path is rotated to an "indicative angle" to ease the recognition and the match process. [Wobbrock 07].

Using a reference angle for each gesture/template, the algorithm resolves the rotation invariance problem;

Scale and Translate : the gesture/template is reduced to a *reference square* thus easing the comparison of the recognition step;

Find the optimal angle for the best score : a distance is computed for the gesture and previously stored templates. The candidate is the closer template (that is, the one with a minimal distance).

The $1 recognizer has been evaluated experimentally in terms of recognition rate and users experience and compared to more complex solutions. Although it satisfies the requirements specified above, the $1 recognizer, as any simple technique, has limitations: because $1 is tolerant to speed, rotation, scale, and position, *it cannot distinguish gestures whose identity depends on specific orientations* (left-to-right or right-to-left arrows), *aspect ratios* (rectangle vs. square), and *location*. In addition, $1 cannot recognize simple 1D gestures, such as lines, nor distinguish gestures based on the speed they have been

stroked with. It cannot either propose dynamic guide since it is based on a post analysis algorithm.

The take away message from the $1 recognizer, is to align with the requirements of $1 for its simplicity. There is still a need for a simple but powerful gesture recognizer capable of detecting both simple and complex gestures while offering users with complete control over the interaction. In particular, as discussed previously, self-reveal, dynamic guidance, and reversibility are key to the usability of gestural menus. Although template-based recognition is a good way to go, single-template is preferable to multiple-sampling-template in order to avoid users trials. These observations have served as the driving principles for the objectives of GeLATI.

7.2. Objectives and Approach

In general, existing implementations of gesture recognizers propose an "eager" version of their offline statistical comparator. For GeLATI, I propose a gesture recognizer that is able to offer the same characteristics as statistical approaches but without the statistical apparatus thus speeding up the recognition process and being a priori "eager" (or *on-line*). The objective is to drastically shrink the number of features required for gesture recognition, thus simplifying the algorithm and speeding up recognition. As for the $1 recognizer, the objective is to be resilient to variations in sampling. By contrast with $1, the goal is to support rotation variance which is an important feature in human gestures. Scale and position variance if necessary (as defined by the developer) should be supported as well. Finally, the recognizer must be able to dynamically propose candidate gestures during the interaction itself not just when users have completed their stroke.

The solution I propose for GeLATI is motivated by the importance of shape as stressed by Rubine and highlighted by Bau *The description of a shape in Rubine's algorithm involves features such as cosine and sine of the gestures' starting angle or distance between the first and last point of the gesture. In the user's perspective, the specification of a gesture is based on the rough appearance of the shape* [Bau 10]. GeLATI proposes a vectorial approach for modeling gestures shape. Figure 7.3 illustrates the approach where a question mark gesture taken from the unistroke set is being "vectorialized". The first shape on the left of the figure represents the ideal path users should follow. The starting point is characterized by a small black circle attached to one extremity of the path. Next to the ideal path, is represented a sampled gesture. Point density may

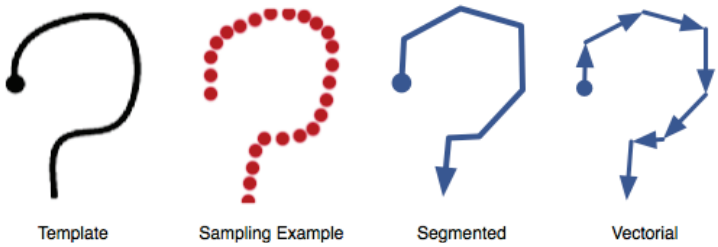


Figure 7.3.: A vectorial approach to gesture recognition. From left to right, the "vectorialization" of a gestural template.

vary according to the speed the gesture is being stroked and the characteristics of the input device used. The third shape represents a segmented version of the ideal path. That is the ideal template represented by a concatenation of straight lines. Then, the segments that compose the third shape are substituted with vectors to build the vectorial representation of the ideal shape. It is composed of a serialization of well defined vectors characterized by a starting point, orientation, direction and magnitude.

The vectorial shape of a gesture is the model used to drive the GeLATI gesture recognizer. Complex gestures are built using a sequence of concatenated vectors.

7.3. GeLATI templates

In GeLATI, gestures templates are represented as a serialization of vectors. On every input event, the recognizer verifies if the designated point is compatible with the current vector or with the next one (if it exists). In case of compatibility, the gesture is confirmed as a candidate for the interaction. Otherwise, the gesture is excluded.

By following the path defined by the vectors the user is able to complete well-defined gestures. However, following a given path is not a simple task, even in direct manipulation using touch screen enabled devices [Accot 97]. Indeed, the actual path deviates from the ideal one. As a result, it is contained within a tunnel rather that being a strict sequence of straight lines. Identifying the ideal width of the tunnel is not straightforward. In an early version of the GeLATI recognizer, preliminary tests did not lead to a clear ideal value. Empirical evaluations suggested a well accepted tunnel width average value around 1cm (that is, 0.5 cm on the left of the vector and 0.5 cm on its right). Nevertheless, having a straight rectangular tunnel around the vector did create some problems. The task of following the tunnel revealed to be too difficult after a certain distance from the activation point. Using a fixed average value for the tunnel width did not appear to be a

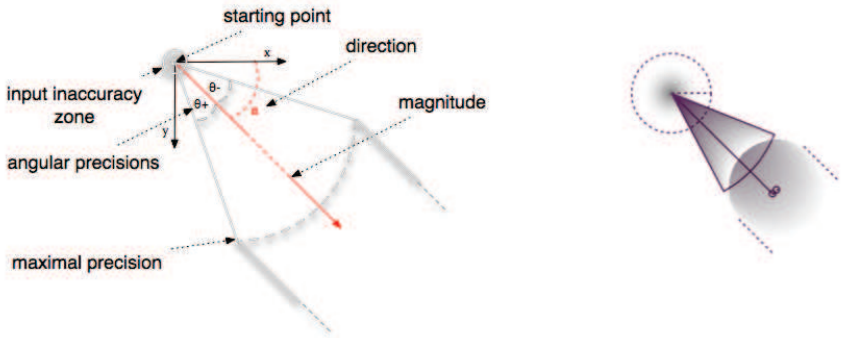


Figure 7.4.: On the left, a simple gesture composed of a single trait: for the simplest cases, an angle (α) is the only parameter needed (direction) to define a gesture of infinite length (the vector magnitude). On the right, a screenshot of a vector with all the GeLATI's parameters represented graphically.

valid choice. To work around the steering law, I designed a compound tunnel composed of a conic section followed by a fixed-width section as shown in Figure 7.4.

The values of the vector parameters depend on the input device as well as on human capabilities. Figure 7.4 On the left, the parameters of a GeLATI vector. In this example, the gesture is an oriented straight line defined by a direction measured by the angle (α), and optionally by a magnitude. When the magnitude is not defined, the gesture is supposed to have an infinite length. The direction is fixed, going away from the starting point. Around the starting point, an input inaccuracy zone is defined to address the inaccuracy of input devices. Touch screens are accurate enough to permit the selection of well-defined points, but they imply an inaccurate user interaction due to the finger moving around the desired point. Angular precision defined asymmetrically with two different θ ($\theta+$ and θ-), permits the task difficulty to remain constant when moving from the starting point toward the desired direction. $\theta+$ and θ- define the breath of the conic section of the tunnel. To avoid the precision to grow up indefinitely, developers can specify a maximal precision which corresponds to the desired maximal tunnel width. Once determined, the size of the tunnel remains constant.

The right side of Figure 7.4 shows an implemented version of the vector presented on the left. The screenshot represents a simple GeLATI gesture. The starting point is characterized by the inaccuracy zone circle drawn with a dotted line. Input variations around the starting point in the inaccuracy zone are ignored. The angular precision zone starts from the starting point toward the gesture direction. After the angular precision, a maximal precision is defined edging the tunnel with a fixed width. Two very small circles are visible near the end of the ideal vector. The first, attached to the vector, represents

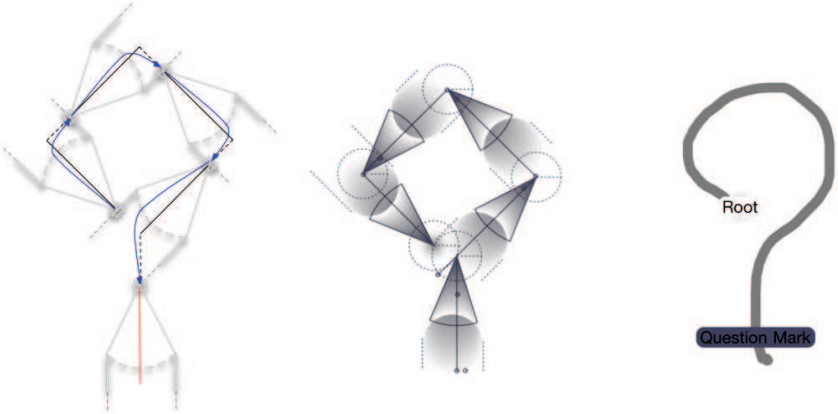


Figure 7.5.: On the left, a GeLATI "Question Mark" as a series/concatenation of five vectors. In the center, a screenshot of the question mark gesture along with the feedback provided by GeLATI. On the right, a screenshot with a different feedback that shows the name of the command (help) as well as the exact trace of the user input.

the ideal user position given the angle of the gesture trait. The second represents the actual user-defined position.

Complex gestures are composed from simple gesture traits that are attached as children of an existing trait. Figure 7.5 shows the "Question Mark" gesture using GeLATI. On the left, a conceptual question mark is proposed as a hierarchy of five vectors, each one (but the last) with a child. In the center, a screenshot of the actual implementation. On the right, a screenshot of the same implementation of the GeLATI "Question Mark" but with a different graphical feedback/feedforward. Instead of showing the algorithm parameters, the implementation proposes a marking menu-like GUI. The command label continuously indicates the direction of the current and next trait. By continuously updating the label position, the user is driven during the interaction. An ink trace marks user input thus showing the whole gesture shape. Here, the user did draw a question mark by simply following the dynamic guidance rendered graphically.

7.3.1. Structure, Aspect and Interaction

From Shneiderman's menus classification, we have learnt the importance of menus layout and of their visual appearance 6.2.1. From Bailly's menus classification, we have noted the importance of interaction 6.2.2. These key recommendations have been taken into account in GeLATI in terms of Structure (layout), Aspect (graphical representation) and Interaction. These three dimensions (Structure, Aspect, Interaction) are not attached to

the gesture as a whole, but at a fine grained level, that is to each vector that composes a gesture. I call these vectors, traits. By doing so, GeLATI is able to support incremental dynamic layout, representation and interaction.

Trait Structure

The trait structure contains the vectorial information together with GeLATI proper characteristics of the trait. Here, the direction angle and the length of the trait are specified together with the diameter of its inaccuracy zone, its angular precision and its tunnel maximal width.

Every input event is passed to the trait structure in order to update the state of the trait. When created, the trait structure is in the *Ready* state. When the user starts an interaction (or when the interaction reaches the interested trait from its parent trait), the trait structure becomes *Active* and starts consuming input events. When the user goes beyond the minimum required length from the starting point while lying inside the trait tunnel, the trait structure becomes *Validated*. When the trait is not interested in input events (e.g., the input event corresponds to a point that lies outside of the tunnel), it tests the point with its siblings. If there is at least a sibling interested in the input event, the structure trait becomes *Forwarding*, otherwise the structure trait enters the *Sleeping* state. If the user reaches the previous level of the hierarchy, the structure trait becomes *Restored*, then Active again.

Trait Interaction

The interaction component of the trait contains the information about the user interaction that characterizes the trait: when to fire the gesture event, i.e. at the end of the interaction or after passing a certain distance from the starting point; whether it has to continuously fire as soon as the trait enters the Validated state, or if it needs to fire once at the end of the interaction (thus reproducing the control menus interaction style). Developers can specify whether the gesture can be considered completed at every trait or if users need to reach the last trait of the chain that composes the gesture. The trait interaction contains the user's speed at runtime as well as a pointer to the method the library has to call as specified by the developer. Different actions can be defined by developers according to the speed the gesture has been stroked with.

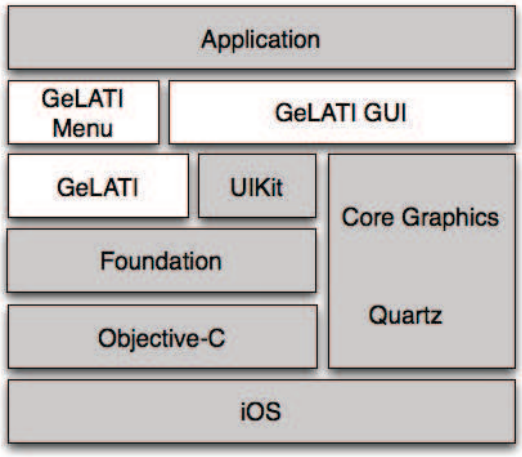


Figure 7.6.: GeLATI is implemented in Objective-C on top of the iOS Foundation Framework. Default GeLATI GUIs are developed over the GeLATI Core using the Core Graphics and Quartz frameworks. A GeLATI menu is built over the GeLATI core and organizes multiple gestures through an entry point hierarchical root. Applications built on top of GeLATI receive the method specified by in gesture definition.

Trait Aspect

The trait aspect contains information regarding how to draw each trait of the gesture such as the preferred color, the cursor or the background images to be used for rendering.

For each gesture, developers specify a tree of GeLATI traits and for each trait, its structure, interaction, and aspect. This is done through a GeLATI Bundle. A GeLATI bundle is a directory that includes three XML files ("structure.plist", "interaction.plist" and "aspect.plist") and a "resources" folder. Files are used by the library when registering a bundle in order to load a GeLATI gestural menu. The files that correspond to the Question Mark gesture described in Figure 7.5 are presented in Appendix A.2.1, Appendix A.2.2, Appendix A.2.3.

7.3.2. Software Architecture

As shown in Figure 7.6, the GeLATI library is written for the iPhone/iPod/iPad on top of the iOS Foundation framework using Objective-C. The GeLATI core is mainly in charge of the organization of the traits that compose a GeLATI menu hierarchy. The GeLATI hierarchy is exposed as a menu component thanks to the "GeLATI Menu" wrapper. The GeLATI menu permits new listeners to be registered (unregistered) and new gesture

sets to be added (removed). Two defaults graphical user interfaces are included in the GeLATI library. They are built on top of the GeLATI core and make use of the Apple UIKit, Core Graphics and Quartz frameworks to best integrate with the iOS graphical environment. The first graphical interface offers a detailed graphical representation of the functioning of the underlying algorithm as well as of the current status of the GeLATI core (as illustrated in Figure 7.4). The second graphical interface proposes an OctoPocus-like interface that simplifies the view for non expert users (as shown in Figure 7.8). More interfaces can be implemented and used as shown in Figure 7.5 where a marking menu-like GUI has been developed. To better analyze the working details of the algorithm we now explain in depth the GeLATI Core and the event management.

The GeLATI Core

The GeLATI algorithm is implemented by three main classes: UMMenu, UMGestureNode and UMStructureTrait (see Appendix A.1 for an overview of all the GeLATI classes).

The UMMenu class, as already introduced, is a menu wrapper to the GeLATI library. It registers/unregisters listeners and gesture bundles.

The UMStructureTrait class implements the GeLATI structure for each trait. The importance of a trait structure has been explained above as it represents the algorithm engine. Here the geometrical and mathematical analyses interpret input events and update the state of the trait (Ready, Active, Validated, Sleeping, Restored) as explained above. Trait status is used by the UMGestureNode to update the gesture tree and to dynamically retrieve candidate gestures for a given interaction.

The UMGestureNode class builds the tree hierarchy that corresponds to the menu structure. A GeLATI menu always starts with a “root” gesture node with no parent and several gesture sibling nodes that correspond to the available gestures. Each gesture node has a structure, an interaction and an aspect object that respectively implement the previously described components of a trait. When an input event is received by the GeLATI menu (UMMenu), this event is hierarchically dispatched to all the traits of the menu (starting from the roots UMGestureNode). The root node dispatches the event to its siblings directly since it has no real trait associated to it. When a sibling node receives an input event, it asks its associated trait to analyze it. The node updates its status according to the status returned by the associated trait.

A gesture node can be:

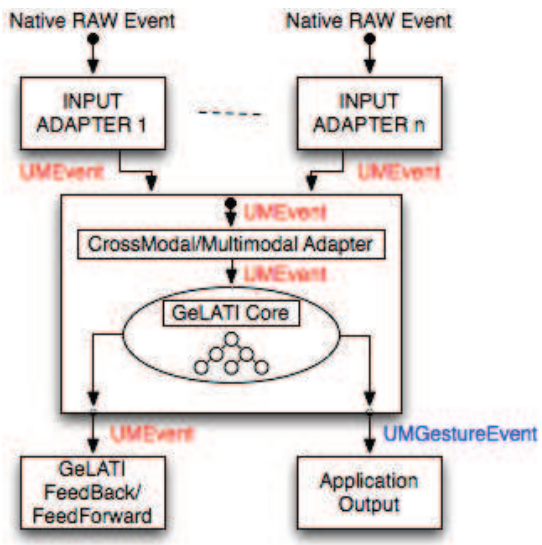


Figure 7.7.: Raw events are transformed into UMEvents and passed to the GeLATI core. GeLATI consumes the events by updating the traits that compose the menu tree. UMEvents are passed to the feedback/feedforward views together with the tree in order to update the aspect graphical rendering. UMGestureEvents are generated and sent to the application to update its status and/or to fire the developer's defined actions.

Ready when it is first created and its trait status is Ready. Or when its trait is "Empty", i.e. it is a root node;

Active when its trait is consuming events;

Idle when its trait is Sleeping.

When the trait has been validated and the interaction object confirms that the gesture is completed (i.e. candidate to the interaction), the gesture node enters the **ChainCompleted** state. This state is a particular Active state. When the node is Active and the structure trait is Forwarding, events are dispatched to the siblings, otherwise sibling nodes just return to optimize the algorithm computation.

The GeLATI Events

Touch and accelerometer based raw events are transformed by input adapters into GeLATI compatible events. According to the input adapter used, touch events can be used alone or complemented with accelerometers data. Once ready, events are passed to the GeLATI core that updates the gesture status and the associated GUI Aspects. Then, the GeLATI core fires the application methods if the gesture is Started, Continued, Ended or has

been Cancelled. Two different kinds of events are used to better synthesize the needs of a gestural architecture:

UMEvent A dedicated event object has been created in order to completely control event management within the GeLATI components. Indeed, UMEvents are input device independent thus permitting a better CrossModal/Multimodal fusion of touch and accelerometer based raw inputs. Independently of the input devices that generated them, UMEvent drives the GeLATI core. Interested nodes of the current menu tree receives and analyzes the current event thus updating its status and the status of its sibling if necessary. Once the tree is updated, UMEvents are passed to the GeLATI GUIs together with the current GeLATI tree. GUIs objects update the GeLATI representation according to the tree status and the received events.

UMGestureEvent GeLATI compatible application receives UMGestureEvents from the GeLATI menu. These high level events encapsulate information about the gesture state, i.e. whether whether a gesture has began, continued, ended or has been cancelled. Applications can then update their status and fire the developers defined actions without considering the details and mechanisms of the GeLATI library.

Having presented the principles and detailed functioning of GeLATI, the following examples show how the power of the library can be exploited.

7.4. Examples of Use

The section illustrates four ways of exploiting GeLATI: (1) Implementing reference menus with GeLATI; (2) Integrating GeLATI in legacy toolkits; (3) Implementing multimodal and crossmodal interactions with GeLATI; (4) Exploiting multiple instance of a GeLATI menu through parallel interaction.

7.4.1. Implementing reference menus with GeLATI

Figure 7.8 shows an overview of the developed GUI aspects.

The two aspects proposed at the top of the figure are the default aspects provided by the library, while the aspect proposed at the bottom has been developed as an external module.

Figure 7.8.: Different GUI aspects can be associated to the same gestural menu. The first raw shows the two default GUI aspects integrated in the library. The first one is an accurate representation of the functioning of the underlying algorithm. The second one is an OctoPocus-like view of a GeLaTI menu. The second row shows a third GUI aspect designed to imitate the Marking Menu.

The top left GUI aspect has been presented above in Section 7.3

The top right GUI aspect is an OctoPocus-inspired graphical representation of the gestural menu. As discussed in the previous chapter, the key contribution of OctoPocus is a continuous feedback and feedforward information flow. On the other hand, OctoPocus has hierarchical and interactional limitations: it does not provide the developer nor the final user with a simple hierarchy management. Nor does it provide a complete exploration mechanism except to start from the beginning. By contrast, this version of OctoPocus allows users to go back and correct their choice incrementally. This is made possible by the GeLATI model of a gesture as a chain of traits where each trait has its own "structure-aspect-interaction" decomposition.

The third example is a marking menu-inspired graphical representation of the same gestural menu where the aspect component has been modified while keeping the structure and interaction identical to the two other examples This has been made possible by the GeLATI architecture.

GeLATI gestures can be defined by exploiting the length of each trait. Figure 7.9 shows an example which, as discussed in Section 7.1, cannot be implemented with $1 recognizer. It is a single trait gesture where three actions are associated to three different lengths. When the user starts the interaction, the “Corto”[2] gesture is represented with the defined fixed length. When the user reaches the end of the first gesture, the second

[2]Corto is the italian for Short.

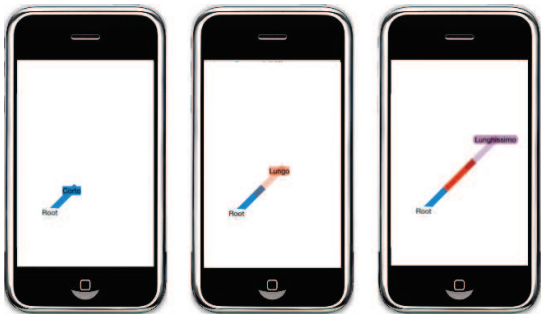


Figure 7.9.: Three gestures are associated to the same vector with three different lengths. An example cannot be supported by the $1.

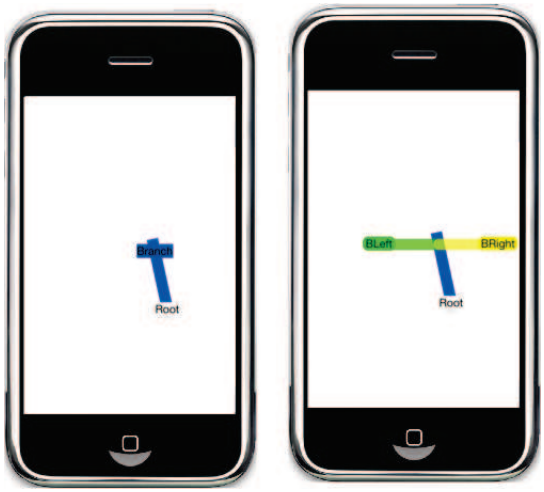


Figure 7.10.: The "Branch" gesture has a fixed length and two siblings: "Branch Right" and "Branch Left".

"Lungo"[3] is represented. The user can choose either to continue for Lungo or to confirm for Corto. The same interaction is repeated for "Lunghissimo"[4]. The Lunghissimo gesture has no well-defined length. In this example, the GUI aspect proposes squared edges for the trait and the command label when a fixed length is defined. Rounded edges characterize gesture traits with undefined magnitude (i.e. free length gestures).

The GeLATI architecture proposes a simple hierarchy management of gestural menus. Figure 7.10 shows a simple example of gestures hierarchy. The first level of the hierarchy is characterized by the "Branch" trait. As shown in the figure, the trait has a fixed length. Once the "Branch" trait validated, the GUI aspect proposes two traits siblings: "Branch Left" and "Branch Right". As shown by the rounded edges, the second level sibling is free length.

[3]Lungo is the italian for Long.
[4]Lunghissimo is the italian for Longest.

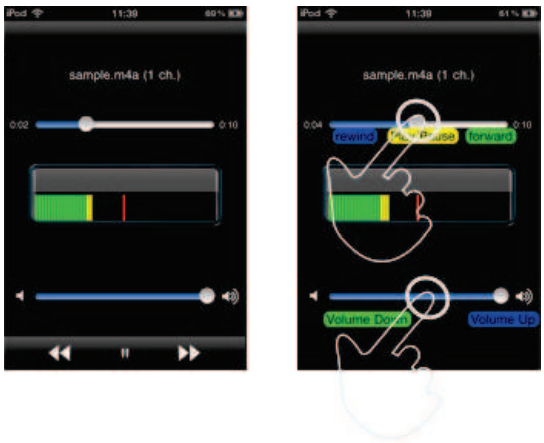
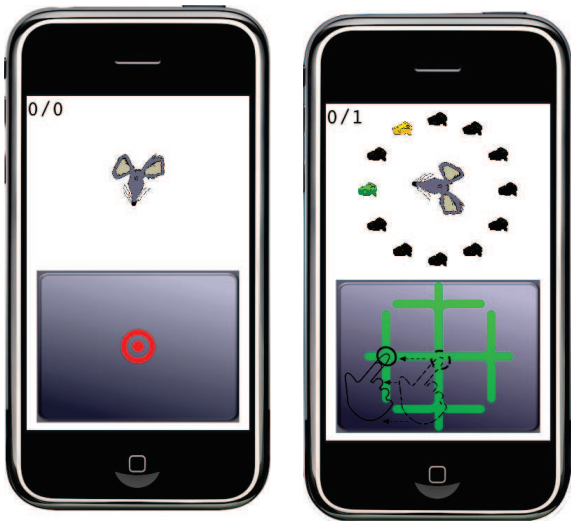


Figure 7.11.: GeLATI gestural menus can be integrated into legacy widgets. Enhancing classical GUI components with gestural interaction permits a complete gestural experience and optimized screen space management.

7.4.2. Integrating GeLATI in legacy toolkits

Figure 7.11 shows a simple example of GeLATI augmented UIKit sliders. On the left, a demo example proposes a simple media player. At the top, a slider indicates the current play position with respect to the whole track. In the center, a sound volume indicator. Under the volume indicator, a slider to control the sound volume. At the bottom, a toolbar contains the usual rewind, play/pause and feedforward buttons. On the right side of the figure, a GeLATI enhanced interface example where the sliders default control has been disabled. The two sliders are enhanced with two simple GeLATI gestural menus. The toolbar at the bottom has been eliminated thus freeing screen space. Three gestures are proposed for the track slider. The first is the root gesture. By simply pressing and releasing the slider, the play/pause command is invoked. Moving to the left fires the rewind command, moving to the right fires the forward button. Here, the invoked methods are speed controlled. The rewind/forward speed is mapped to the gesture speed. Two gestures are proposed for the volume slider. The GeLATI menu is simply invoked by touching the slider. Moving to the left decreases the sound volume while moving to the right will high it up. In this case, the menu is position controlled. The slider is moved proportionally to the magnitude of the gesture.



(a) (b)

Figure 7.12.: Two screenshots showing the test application and the interaction I designed to test the accelerometers+touch screen synaesthesia. In 7.12(a) the application is waiting for a new interaction to start. In 7.12(b) the interaction started and continued selecting the cheese on the left through a finger movement. The task of the active trial is to select the yellow upper-left cheese, while the currently selected cheese is shown in green. User could either control the application through the touch screen or though the accelerometers.

7.4.3. Multimodality with GeLATI

Three input modules are integrated in the GeLATI library by default. A touch based one, and two accelerometers based, the first position controlled and the second speed controlled. A third module has been created in order to analyze the touch+accelerometers "synaesthesia" (c.f. Chapter 2) using the GeLATI library and experimented in an informal test application visible in Figure 7.12(a) and Figure 7.12(b).

Previously conducted experiments showed us the difficulties users have in interacting with a menu using traditional accelerometers-based techniques (position and speed controlled). Acceleration-based tilting control made users concentrate on the control itself and created problems in stopping the interaction. Position-based tilting techniques need interpolation filters to stabilize accelerometers values. Filters introduced delays making the interaction less responsive.

I implemented a novel tilting technique letting even non-expert users simply interact with the device. The user had to tilt enough the device on the desired direction to overcome a predefined activation threshold. Once the direction was chosen, users had

to orientate the device back on the starting position. The vertical and horizontal axes were independent and corresponded to the pitch and roll of the device. The following algorithm describes the roll behavior users were asked to control. The pitch behavior can be simply retrieved.

Algorithm 1 Tilt algorithm for roll

Require: $delta_roll$,roll_threshold
  if $rool$ **then**
    if $delta_roll > rool_threshold$ **then**
      $roll \leftarrow FALSE$
      $roll_gain \leftarrow$roll_gain+1
    else if $delta_roll < -roll_threshold$ **then**
      $roll \leftarrow FALSE$
      $roll_gain \leftarrow$roll_gain-1
    end if
  else if $abs(delta_roll) < (roll_threshold/2)$ **then**
    $roll \leftarrow TRUE$
  end if

7.4.4. Parallel interaction with GeLATI

Multiple instances of a GeLATI menu can be created while sharing the same graphical aspect. In Figure 7.13, left, a simple white canvas (a UIView) has been enhanced with a GeLATI menu. The user starts two distinct interactions in parallel. Both of them are driven independently thus permitting collaborative exploitation. An example of such interaction is visible on the right side of the figure. Using two parallel interactions, the user is able to position and orient the button widget independently.

7.5. Limitations

GeLATI, as any simple approach, comes with some intrinsic limitations. Other limitations can be overcome easily by improving the library *per se*.

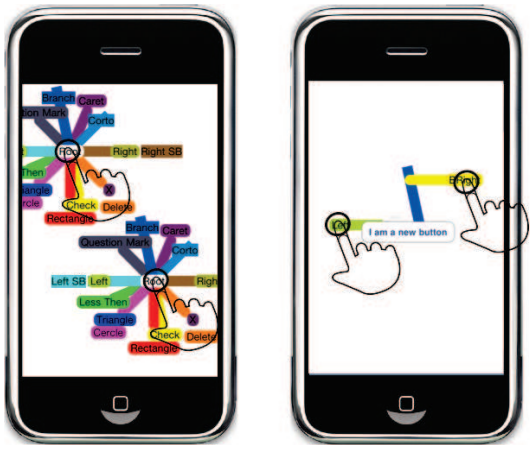


Figure 7.13.: Multiple instances of a GeLATI menu can be activated with the same GUI aspect or with different ones. On the left two GeLATI menus have been activated on the main view. On the right two menus have been activated on the same graphical button. The "BRigth" options controls the button orientation while "Left" option control the widget position.

7.5.1. GeLATI intrinsic limitations

By essence, the GeLATI recognizer does not permit a sibling trait to have the exact opposite direction of its parent trait: (1) Once a trait has been validated, it is ready to forward events to all candidate siblings. (2) Moving up to the node opposite direction is interpreted as a back up the node hierarchy as long as the input event still satisfies the path constraints.

GeLATI proposes an intermediate point of view in between position driven and gestural GUIs. Nevertheless the approach impose some limitation when considering user exploration in menus. In particular when the user move among siblings of the same node (i.e. brothers). GeLATI menus have an acyclic tree structure, thus preventing direct moves among brothers without stepping by the father.

7.5.2. Future work

The library should let the developer choose out to deal with bimanual/multi-fingers interaction to work either in parallel or complementarily. In the current version, multiple instances of a GeLATI menu on the same graphical object cannot compose more complex syntaxes but are interpreted as independent instances. While the proposed library has

been designed with composition in mind, work needs to be done to completely integrate this feature.

The GeLATI library has been studied to permit fast prototyping and testing gestural interactions. User testing has been conducted during the development of GeLATI in order to drive the development and fix (wrong) hypotheses. Nevertheless, the library needs to be tested with other developers.

Finally, a graphical user interface to create GeLATI menus from examples needs to be developed.

7.6. Synthesis

An introduction on existing features based statistical gesture recognizer have been presented. Motivation justifying a non-statistical approach have been highlighted. The need of an on-line hierarchy management; the necessity to improve touch and accelerometers harmony in composing gestural interactions justified the research. This chapter has described GeLATI and its vectorial approach to gesture recognition. In particular I analyzed the GeLATI algorithm according to the seven points introduced at the beginning of the chapters: (1) gesture recognition is driven by a new vectorial approach and combines cross-modal and/or multi-modal interactions; (2) gesture recognition is driven by a single-sample template; (3) gestures are rectilinear but GeLATI reduces users constraints in completing them; (4) GeLATI supports both deictic and semaphoric gestures; (5) GeLATI aspects propose to novice users guidelines and guidance in completing the interaction; (6) a hierarchical management insures the undo-ability of the interaction and supports exploration; (7) GeLATI integrates with legacy toolkits enhancing existing graphical widgets with gestural interaction. In GeLATI features are reduced to the minimum, thus simplifying the task of gestural menus specification and recognition. The GeLATI software architecture has been presented and detailed. Examples have been described together with interaction properties GeLATI menus are characterized by. Limitations and Future work have been presented characterizing the current state of the art of the proposed solution.

Part V.

Conclusions

Chapter 8.

Conclusions

The preceding chapters have shown the importance, diversity, and complexity of gesture-based interaction techniques for hand-held devices. Although gesture is viewed as a natural way to interact with a machine, developing effective gestural interaction techniques is still a challenging task. Designers cannot rely yet on standard conventions such as the interaction patterns developed for WIMP user interfaces. Instead, they have to cope with a perpetual technology push that continuously widens the design space. In addition, the implementation of gesture recognizers along with their integration into current user interface toolkits is not a straightforward matter.

In this context, this dissertation is concentrated on gestural interaction techniques for a specific class of devices (i.e. touch screen and accelerometers-enabled handhelds) with two key concerns: (1) to provide designers with a conceptual framework that structures reasoning about the nature of gesture-based interaction techniques, and (2) to provide developers with an efficient and robust gesture recognizer that can be easily integrated into existing toolkits.

8.1. Contributions

The contributions of this thesis is three-fold: conceptual, with taxonomies for reasoning about gesture and gestures for hand-held devices; technical, with the GeLATI library; and interactional, with the design of novel interaction techniques such as TouchOver.

8.1.1. Gestural Classification

I propose a synthesis of the taxonomies that have been developed for gestures in psycholinguistics as well as in Computer-Human Interaction. Some of them focus on the functions of gesture, others classify gestures according to their morphology while others blur the distinction between function and form. By correlating these perspectives into a single framework, new researchers in the field are provided with a unified concise view so that they can rapidly relate the different approaches, perspectives, and terminologies.

Additionally, this framework is used to clarify the coverage and focus of my own interest: that of deictic and semaphoric gestures for computer human interaction. With regard to expressive power, I address gestures that cover the specification of a single token up to the expression of commands along with their parameters.

A synthesized taxonomy for gestures is appropriate for comprehending the breadth of the domain, but not for reasoning at a fine grain about the design of a gesture-based interaction technique. For this purpose, I propose a new taxonomy for gestural interaction techniques based on accelerometers.This taxonomy, which is motivated by the foundational contributions in HCI, brings together Foley's generic tasks with the formal lexical, syntactic, semantic and pragmatic dimensions of languages to characterize the physical actions involved in gestural interaction. It is radically centered on human physical actions. The hypothesis is that the physical action is the appropriate atomic level from which novel interaction techniques can be designed to provide system-wide consistent languages. In this taxonomy, the abscissa defines the lexicon in terms of the physical manipulations users perform with the device, with a clear separation between background and foreground interaction. The ordinate corresponds to Foley's interaction tasks. An interaction technique is plotted as a point in this space where each point is decorated with pragmatic and syntactic properties. The taxonomy includes two syntactic modifiers: whether the interaction technique is clutched or continued, and the control type (in position, speed, or acceleration). One pragmatic modifier indicates the degree of indirection of the interaction technique.

In order to demonstrate its flexibility and coverage, this taxonomy has been applied to the classification of post-WIMP accelerometers-based interaction techniques as well as to WIMP classical interaction techniques.

Among existing interaction techniques, menus play a prevalent role. For this reason, I have reviewed the design spaces developed for menus as well as the definitions and the

subtle terminology distinctions between widget, interactors, and interaction instruments. From there, I propose a clear distinction between software instruments and hardware instruments as well as a revised version of Bailly's definition of menus.

A gesture is a physical instrument whose hardware component is a direct input device (mouse, pen or touch screen) and the software component, a gesture recognizer.

A menu is a software instrument that satisfies the following requirements:

1. It supports the selection of one item within a finite set of options.
2. It minimizes mental efforts by presenting the set of options to the user.
3. It presents the options as a semantically and spatially meaningful structure.
4. It is transient in the sense that it is perceivable to the user only during its interaction with the user.
5. It is quasimodal since it defines a local context for system interpretation that is maintained explicitly by the user until the interaction ends.

.

A gestural menu is a menu software instrument connected to a gesture instrument such that the selection process is driven by gestures.

8.1.2. GeLATI

GeLATI is a library to design, prototype and implement gestural menus.

The key features of GeLATI are the following:

1. Gesture recognition is based on a new real time vectorial approach that combines multiple inputs to support cross-modal and/or multi-modal interactions;
2. Gesture recognition, which uses a single-sample template approach, does not need training;
3. Gestures, which are modeled as series of vectors, are rectilinear. (Rectilinear gestures have been demonstrated as been more usable than free form gestures.);
4. Both deictic and semaphoric gestures are supported;

5. Novice users are guided along well-defined paths to complete a gesture stroke. Expert users can stroke commands without graphical feedback as exemplified by the Marking Menus;
6. Exploration is supported through an incremental back-to-hierarchy-node mechanism;
7. Hierarchical gestural menus can be automatically integrated into existing graphical toolkits.

In short, the GeLATI recognizer is able to offer the same characteristics as statistical based approaches but without the statistical apparatus thus speeding up the recognition process. As for the $1 recognizer, the recognizer is resilient to variations in sampling. By contrast with $1, the goal is to support rotation variance which is an important feature in human gestures. Scale and position variance if necessary (as defined by the developer) are supported as well. Finally, the recognizer is able to dynamically propose candidate gestures during the interaction itself not just when users have completed their stroke.

A number of examples have been implemented to demonstrate the functional coverage of GeLATI as well as its integration into existing graphical toolkits.

8.1.3. TouchOver

TouchOver is a complementary multimodal input for one hand interactions on touch-screen based accelerometers-enabled handheld devices. TouchOver offers a three-state input model similar to the stylus tablet input with two states where the system tracks the finger's motion, thus adding a passive tracking state to touch input. This creates new opportunities for handheld device interaction techniques like on-over interactions, feedforward, or visual and eye-free user interface exploration.

The proposed interaction technique has been experimentally validated in precision and speed.

8.2. Limitation and Perspectives

Two main limitations need to be addressed in the GeLATI framework: (1) the re-selection mechanism and (2) the integration of non rectilinear gestures.

8.2.1. The re-selection mechanism

When a user interacts with a GeLATI menu, he/she crosses several hierarchy nodes by simply following the defined visible path. When correcting his/her choice, the user goes back the interaction path. In a position-based menu, the user position is simply tracked and analyzed with respect to the graphical object bounding box. In a GeLATI-based menu, the user position is checked with respect to the latest point in each trait that composes the gesture. An interesting analysis could be to test the user's satisfaction and to validate the new approach in order to integrate the algorithm in other existing gestural recognizers. Another way is to introduce the bounding box concept in GeLATI, thus merging position-based and gesture-driven approaches.

8.2.2. Non-rectilinear traits

GeLATI algorithm is based on the main hypothesis that a sequence of linear traits (vectors) composes complex shapes thus driving users through the gestural interaction. An interesting scenario should be evaluated extending the basic modules composing complex gestures also to non-linear traits. For example, a gesture could be defined as a sequence of a $y = sin(x)$ trait plus an $y = x$ trait rather than the simple vectorialization algorithm proposed.

8.3. I should have. . .

The applicability of the GeLATI approach has been tested in several informal experiments and through different demonstrators. Nevertheless, the proposed architecture and library still need to be tested with other programmers and designers. In addition a formal user study needs to be designed in order to verify users acceptance in integrating gestures into classical widget components.

Synthèse en français

Résumé

Dans cette thèse, j'aborde la question de l'interaction gestuelle sur dispositif mobile. Ces dispositifs, à présent communs, se distinguent des ordinateurs conventionnels principalement par leurs périphériques d'interaction avec l'utilisateur (écrans de taille restreinte mais tactiles, capteurs divers tels que les accéléromètres) ainsi que par le contexte dans lequel ils sont utilisés. Le travail que je présente est une exploration du vaste domaine des techniques d'interaction sur ces dispositifs mobiles. Je structure cet espace en me concentrant sur les techniques à base d'accéléromètres pour lesquelles je propose une taxonomie. Son pouvoir descriptif et discriminant est validé par la classification de trente-sept techniques d'interaction de la littérature. La suite de mon travail se penche sur la réalisation de techniques d'interaction gestuelles pour ces dispositifs mobiles. Avec TouchOver, je montre qu'il est possible de tirer parti de manière complémentaire de deux canaux d'entrée (écran tactile et accéléromètres) pour ajouter un état au glissé du doigt, permettant ainsi d'enrichir cette interaction. Enfin, je m'intéresse aux menus sur dispositif mobile et je propose une nouvelle forme de menus gestuels. Je présente leur réalisation avec la bibliothèque logicielle GeLATI qui permet leur intégration à une boîte à outils de développement d'interface graphique préexistante.

Introduction

Introduction

Contexte et motivations

Cette thèse s'intéresse à la conception et au développement des techniques d'interaction basées sur le geste pour dispositifs mobiles. Les appareils mobiles sont souvent présentés comme des ordinateurs de bureau dotés de faibles capacités de calcul et une bande passante restreinte en entrée/sortie. Dans cette thèse, je pars du point de vue que les appareils mobiles sont fondamentalement différents des ordinateurs de bureau, et donc la métaphore classique des ordinateurs de bureau leur est inappropriée. Ce point de vue est motivé par deux observations. Tout d'abord, les ordinateurs de bureau et les appareils de poche sont des solutions pour des contextes d'utilisation différents. Ensuite, ils offrent des caractéristiques d'interaction fondamentalement très différentes.

Les smartphones tels que ceux présentés dans la Figure 8.1, les lecteurs audio et les tablettes PC ont désormais atteint des performances élevées tout en conservant une taille relativement compacte. Ce qui permet aux utilisateurs d'effectuer leurs tâches lors des déplacements, sans les contraintes de PC de bureau. Contrôler l'e-mail ou naviguer sur le web dans le bus, écouter la musique pendant le footing, ou encore garder trace de rendez-vous où et quand nécessaire, sont des exemples typiques de scénarios de la vie quotidienne.

Par le biais de technologies de capteurs peu coûteux intégrés dans les appareils mobiles, les compétences et habilités humaines peuvent désormais être capitalisées de façon innovante. En particulier, une toute nouvelle gamme de possibilités d'interaction physique basée sur les compétences humaines en manipulation a été ouverte. Au lieu d'interagir avec des ordinateurs *à travers* des dispositifs physiques tels que les souris et les claviers, l'interaction peut se produire *avec* le dispositif mobile lui-même. Peut-être, la meilleure illustration de cette tendance est le concept d'«interfaces utilisateur incarnées» [Fishkin 98] ou d'«interfaces utilisateur manipulatrices» [Harrison 98] qui, à leur tour, sont des approches spécifiques pour les «interfaces utilisateur tangibles» [Fitzmaurice 93, Ishii 97].

Figure 8.1.: De gauche à droite : une capture d'écran de Microsoft Windows Mobile 7, montrant l'interface d'une animation en perspective ; l'écran d'accueil d'un iPhone d'Apple, l'interface de sélection des tâches dans le Palm Pre.

L'importance des gestes

Les gestes sont inextricablement tissés dans nos vies. Citant Axtell : «Sans les gestes, notre monde serait statique, incolore ... Mario Pei, un expert en communication, une fois a estimé que les humains peuvent produire jusqu'à 700 000 signes physiques différents. Birdwhistell estime que le visage seul est capable de produire 250 000 expressions et rapporte que le chercheur M. H. Krout a identifié 5000 gestes distincts de la main» [Axtell 91]. Ces résultats quantifiés indiquent que les gestes peuvent jouer un rôle significatif dans l'Interaction Homme-Machine (IHM). Ils expriment aussi l'immense difficulté rencontrée par les chercheurs IHM pour concevoir et développer des techniques d'interaction basées sur les gestes qui soient efficaces.

Selon Baudel, les techniques d'interaction basées sur les gestes sont efficaces dans le mesure où ils peuvent être concis, naturels et directs [Baudel 95, Norman 10, Morrel-Samuels 90] :

Concis quand ils permettent aux utilisateurs de spécifier à la fois une commande et ses paramètres en une action atomique.

Naturel quand ils correspondent à l'attente des utilisateurs au regard de la commande qu'ils appellent.

Direct quand ils permettent des manipulations directes vraiment directes, sans passer par «l'indirection» de dispositifs d'entrée physiques intermédiaires.

D'autre part, «les systèmes gestuels ne sont pas différents de toute autre forme d'interaction» [Norman 10] dans ce sens qu'ils doivent suivre les règles de base en conception d'interaction : typiquement, un modèle conceptuel approprié doit être mis au point, de même feedforward et feedback, ainsi que des mécanismes pour l'interruption et l'annulation de l'action doivent être des préoccupations majeures. En outre, en raison de leur qualité d'être naturels, les gestes peuvent être ambigus et peuvent être adressées par inadvertance au système.

Bien que l'interaction gestuelle ait été étudiée depuis le début des années soixante, nous n'avons pas encore de conventions standard de même nature que les modèles d'interaction développés pour les interfaces utilisateur WIMP. Comme le montrent les exemples ci-dessous, l'interaction gestuelle a abordé de nombreuses directions potentielles donnant lieu à nombre de solutions prolifiques.

Une palette d'interfaces gestuelles

Les exemples suivants illustrent l'ampleur des solutions actuelles. Ils ne sont pas destinés à fournir un aperçu complet de l'état de l'art. Une analyse plus détaillée de l'état de l'art sera présentée au Chapitre 2. Nous observons à peu près quatre groupes de techniques d'interaction gestuelle : gestes en l'air, gestes reposant sur le toucher de surfaces, gestes de manipulation, et gestes combinés avec d'autres modalités.

Les gestes en l'air

Le travail pionnier de Myron Krueger en réalité artificielle au début des années 1980 est peut-être la première introduction de l'interaction gestuelle du corps dans les airs avec de grandes images projetées à l'aide d'une vidéo-caméra pour le suivi complet du corps.

Charade, de Baudel, est la première illustration française de gestes de la main en 3D en utilisant un DataGlove. La Figure 8.3 montre la configuration du système d'interaction Charade ainsi qu'un exemple des gestes de contrôle d'un visualisateurs de diapositives.

Figure 8.2.: Krueger est l'un des chercheurs pionniers des gestes à corps complet.

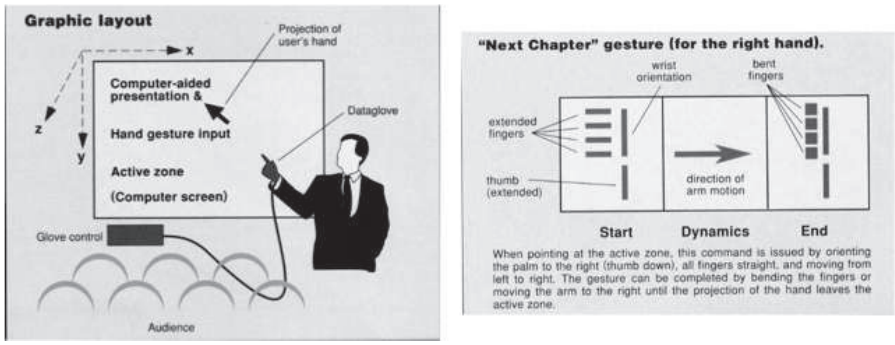


Figure 8.3.: Charade utilise un modèle d'interaction 3D à la main. Sur la droite, un exemple de la notation utilisée pour désigner le geste «chapitre suivant» [Baudel 95].

Gestes tactiles

Le nombre de gestes tactiles sur les surfaces graphiques tactiles a littéralement explosé. En réponse à cette diversité, Villamore *et al.* ont créé un référentiel de gestes afin d'organiser et de classer tous les gestes possibles et leur disponibilité dans des produits commerciaux [Villamore 10]. De plus, Wobbrock *et al.* ont analysé l'utilisation des gestes définis par l'utilisateur [Wobbrock 09].

Parmi ces systèmes, OctoPocus [Bau 10] et MicroRolls [Roudaut 10] méritent une attention particulière. OctoPocus combine une rétroaction immédiate avec l'anticipation de manière fortement couplé. En montrant tous les chemins possibles de façon incrémentale, (1) les utilisateurs savent ce qu'ils ont déjà fait (feedback) et les possibilités qu'ils ont (feedforward), et (2) experts et novices sont soutenus de manière flexible. La Figure 8.5

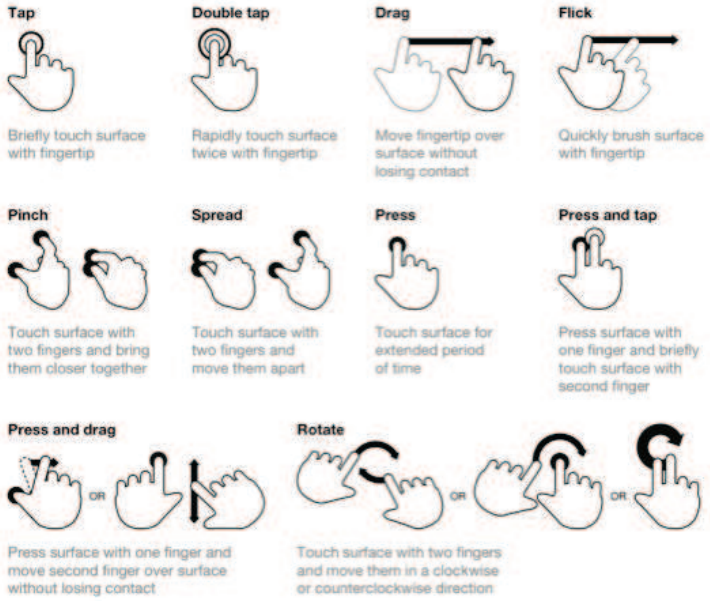


Figure 8.4.: Villamore *et al.* proposent une référence pour tous les gestes tactiles mis en œuvre dans les systèmes modernes tels que iOS, Windows Mobile 7 ou WebOS [Villamore 10].

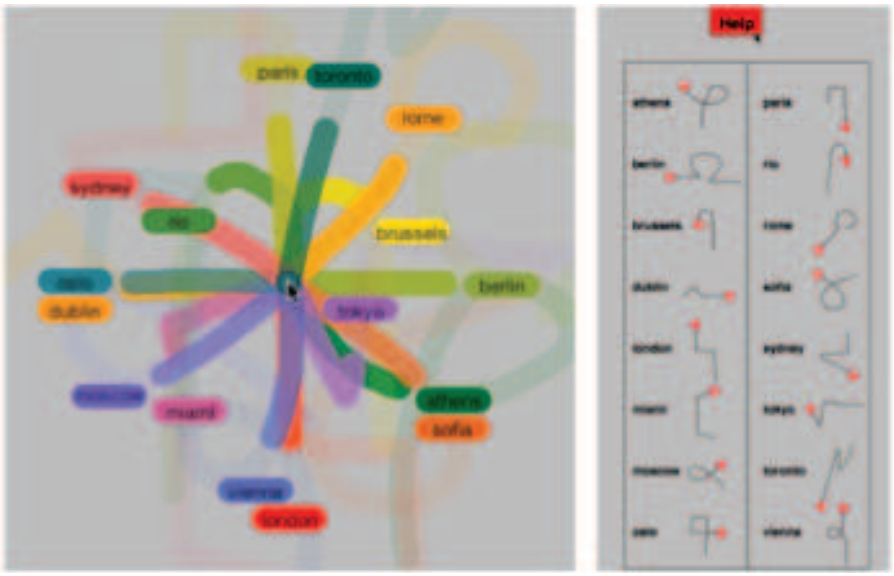


Figure 8.5.: OctoPocus intègre la rétroaction avec l'anticipation dans les interfaces gestuelles pour aider les utilisateurs à découvrir et apprendre les menus gestuels [Bau 10].

Figure 8.6.: Roudaut propose l'exploitation des micro-gestes au pouce sur les appareils à écran tactile avec des «Microrolls» [Roudaut 10].

montre un exemple d'un menu OctoPocus avec les gestes associés.

La technique d'interaction MicroRolls exploite les micro-gestes au pouce comme un mécanisme pour enrichir le vocabulaire d'entrée tout en exigeant une petite «empreinte» sur l'écran. Comme le montre la Figure 8.6, les micro-gestes sont accomplis en penchant le doigt dans six orientations différentes sans la nécessité de déplacer le doigt sur l'écran. Six commandes différentes sont associées aux six MicroRolls distincts. En outre, les MicroRolls peuvent être combinés avec le déplacement du doigt pour proposer une interaction gestuelle plus complète, comme proposé dans le RollMark Menu de la Figure 8.6.

Gestes de manipulation

Les gestes de manipulation tels que le squeeze, l'inclinaison, et le shake, sont appliqués au corps physique de l'appareil même. Avec des gestes de manipulation, *le corps de l'appareil fait partie de l'interface utilisateur*, d'où, le terme «interface utilisateur incarnée» [Fishkin 98]. Comme exemple de geste de manipulation (voir Figure 8.7), l'utilisateur incline l'appareil pour basculer entre les applications précédemment ouvertes activant la modalité par un simple «tap» sur le dos de l'appareil. D'autres travaux caractéristiques (et pionniers dans ce domaine) comprennent [Fitzmaurice 93, Hinckley 00, Levin 99, Partridge 02, Rekimoto 96] qui ont ouvert la voie à un domaine de recherche actif [Ballagas 06, Williamson 07, Wilson 03].

Le geste est aussi une modalité d'entrée, qui, en tant que tel, peut être combiné avec d'autres modalités comme la parole.

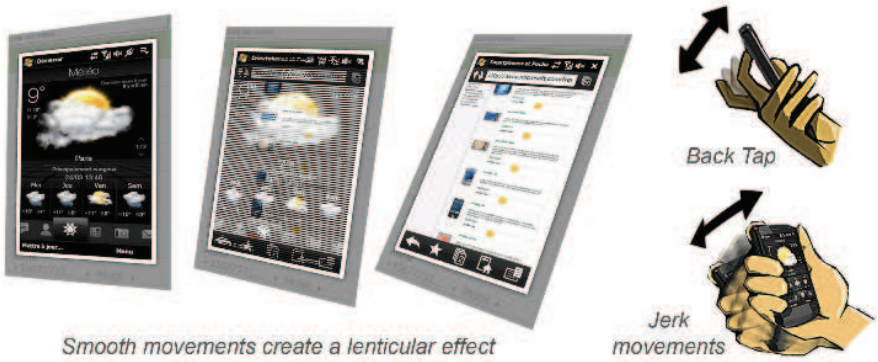


Figure 8.7.: TimeTilt propose un exemple d'interaction multimodale qui exploite les accéléromètres couplés avec deux langages d'interactions différentes [Roudaut 10].

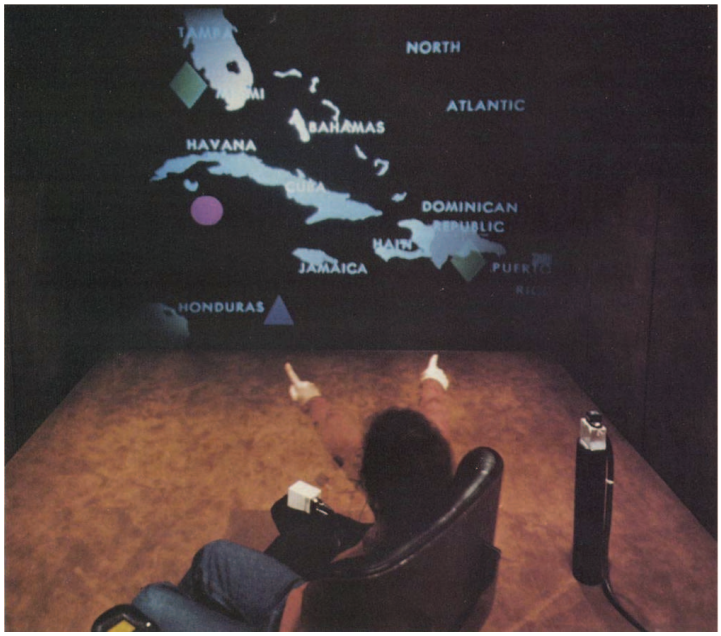


Figure 8.8.: Bolt utilise la combinaison de la voix et des gestes à la main pour commander les applications dans la «Media Room» [Bolt 80].

Interaction multimodale

Bolt et son «met-ça-là» sert de référence paradigmatique pour l'interaction multimodale où la parole et le geste peuvent être utilisés de manière complémentaire pour manipuler des formes graphiques dans les airs (voir Figure 8.8). Typiquement, le geste assure la fonction de déictique comme dans :«Déplacer ça vers la droite du carré vert» ou «Met ça là».

L'interaction multimodale basée sur la parole et les gestes déictiques a été étudiée depuis les années quatre-vingt. Bon nombre de solutions et de boîtes à outils sont maintenant disponibles dans lesquelles l'interaction multimodale inclut la parole, les gestes au stylet, et les gestes en l'air [Oviatt 92, Cohen 89, Chatty 04]. D'autre part, le problème de l'interaction multimodale impliquant les accéléromètres comme dispositifs d'entrée n'a pas été abordé de manière exhaustive. Les capteurs seuls ne sont pas toujours capables de déterminer si une interaction a commencé, continué ou terminé. Une modalité complémentaire offre généralement un moyen naturel à l'utilisateur pour désigner ces marqueurs. Par exemple, Hinckley *et al.* ont exploré la complémentarité synergique du toucher et du mouvement pour les dispositifs mobiles [Hinckley 11].

En outre, les systèmes d'exploitation et les boîtes à outils originaux pour les appareils portables n'ont pas été conçus pour faciliter l'intégration de nouvelles techniques d'interaction. En particulier, l'intégration de nouvelles techniques pour contrôler les widgets préexistants n'est pas trivialle. Dans la plupart des cas, l'architecture sous-jacente est étroitement liée aux propriétés d'interaction WIMP, résultant en de fortes limitations techniques. Par exemple, un pointeur de souris a toujours une position, les comportements des éléments graphiques des interfaces utilisateurs sont donc souvent guidés par la position du pointeur. Un tel, petit, détail devient une limitation considérable lorsqu'on tente de contrôler ces mêmes widgets en utilisant des accéléromètres.

Objectifs du travail

Ce travail a deux objectifs principaux : (1) Organisation de l'espace des menus fondés sur le geste, en particulier ceux qui exploitent les entrées multimodales (écrans tactiles et accéléromètres), (2) Proposition de nouvelles techniques d'interaction gestuelle basées sur l'intégration du toucher et des accéléromètres.

Techniques d'interaction gestuelle

Différentes taxonomies caractérisant les interactions gestuelles ont déjà été proposées dans la littérature de l'Interaction Homme-Machine (IHM). Dans cette thèse, mon premier objectif est de proposer un état de l'art des approches existantes et une catégorisation des techniques existantes.

La synthèse des approches existantes permettra de les améliorer à travers la proposition d'une taxonomie centrée utilisateur grâce à laquelle je pourrai organiser l'état de l'art sur les techniques d'interaction basées sur les accéléromètres.

Mon deuxième but est de parvenir à une interaction transparente utilisant à la fois des écrans tactiles et des accéléromètres afin qu'ils travaillent ensemble à la fois en collaboration et en alternative (Complémentarité et Redondance en termes de propriétés CARE [Coutaz 95]) pour proposer une interaction multimodale .

Les menus pilotés par le geste

Les techniques d'interaction gestuelle sont souvent utilisées pour contrôler des menus graphiques. Afin d'offrir une expérience utilisateur riche, j'ai besoin de surmonter les difficultés spécifiques des interactions gestuelles. Comme décrit par Baudel [Baudel 95], ces difficultés ont encore besoin d'une solution claire et d'une bonne intégration architecturale :

Algorithme d'interprétation. Je cherche une approche générique pour offrir un algorithme capable d'interpréter les gestes de formes différentes sans qu'il soit limité par une phase d'apprentissage.

Caractéristiques des gestes. Je cherche à comprendre quelle classe de gestes est la mieux adaptée pour les dispositifs mobiles et les caractéristiques des écrans tactiles et des accéléromètres.

Choix de l'Interaction. Les travaux existants sur les techniques d'interaction basées sur les gestes présentent deux caractéristiques principales : (1) une interprétation hors ligne de l'interaction et (2) un niveau de hiérarchie unique dans l'organisation des gestes. Mon objectif est de gérer le suivi en ligne et la reconnaissance de gestes et d'offrir la capacité d'organiser ces gestes dans les menus, proposant ainsi la personnalisation des structures du menu de ses niveaux hiérarchiques.

Synthèse des contributions

Je propose un aperçu des approches existantes de techniques d'interaction gestuelle. Différents travaux des 20 dernières années sont organisés afin de se concentrer sur les capacités de communication des interactions gestuelles dans le domaine de l'IHM. Je propose un état de l'art de techniques d'interaction gestuelle basées sur les accéléromètres

pour les dispositifs mobiles. Ces techniques sont organisées grâce à une taxonomie centrée utilisateur innovante. Je présente TouchOver, une nouvelle technique d'interaction pour effectuer la tâche de sélection sur des dispositifs mobiles dotés d'accéléromètres et d'écran tactile. Ma dernière contribution est la bibliothèque GeLATI pour la reconnaissance des gestes. GeLATI met en œuvre un algorithme de reconnaissance vectoriel pour composer des menus avec des gestes rectilignes.

Aperçu de la dissertation

Cette thèse est structurée en cinq parties principales, la première étant cette introduction et la dernière les conclusions finales. Les parties centrales composent le cœur de ce travail et sont décrites ci-dessous en détails :

Part II Décrit l'état de l'art scientifique et industriel des techniques d'interaction basées sur les accéléromètres pour dispositifs mobiles. En particulier :

1. Le premier chapitre étudie la caractérisation des gestes en fonction des taxonomies et des définitions existantes. Plusieurs travaux sont présentés et la portée de ce travail est définie par les classes des gestes que je vais aborder.

2. Je présente les fondamentaux sur les taxonomie de techniques d'interaction et de périphériques d'entrée. Je présente une nouvelle taxonomie pour les techniques d'interaction gestuelles. Je classe certaines techniques d'interaction WIMP bien connues pour mieux clarifier l'utilisation de la taxonomie proposée.

3. Ensuite j'utilise cette nouvelle taxonomie pour classer l'état de l'art des techniques d'interaction basées sur les accéléromètres. Je décris plus de vingt techniques d'interaction qui ont été proposées au cours des vingt dernières années. Je conclus avec une discussion, plus large, du cadre général proposé par la classification et comment elle a aidé à faire progresser ma recherche.

Part III Mes contributions techniques consistent en la proposition de TouchOver, une technique d'interaction pour la sélection et d'une infrastructure pour mettre en œuvre des techniques d'interaction gestuelles multimodales. Cette partie du travail introduit TouchOver : Je propose TouchOver, une technique d'interaction multimodale, qui découple les tâches de positionnement et de sélections élémentaires sur les dispositifs mobiles dotés d'écran tactile et d'accéléromètres. Avec TouchOver, le positionnement est effectué avec un doigt sur la surface tactile, tandis que la

sélection est véhiculée par une inclinaison douce de l'appareil. En faisant ainsi, TouchOver ajoute un état «survol» et améliore la précision de la sélection tout en restant compatible avec les techniques d'interaction existantes.

Part IV montre qu'un ensemble de gestes organisés dans un widget graphique peut composer un menu. Ces menus graphiques gestuels doivent être bien intégrés avec les widgets existants tout en exploitant les caractéristiques d'entrée multiples offertes par le dispositif :

1. Je présente un état de l'art succinct des menus graphiques. J'analyse les définitions existantes ainsi que des exemples afin de mieux cerner le domaine.

2. Cette section est dédiée à la troisième contribution de ce travail : la bibliothèque/architecture GeLATI. GeLATI est une approche vectorielle à la reconnaissance de gestes rectilignes. Les gestes plus complexes sont décomposés en traits. Les traits sont organisés hiérarchiquement afin d'offrir la possibilité de créer des menus gestuels.

Un espace de conception pour les techniques d'interaction gestuelles basées sur les accéléromètres

Caractérisation et classification des gestes

Dans ce chapitre, je propose une synthèse des taxonomies qui ont été développés pour les gestes. L'objectif est de construire une vue unifiée et concis pour un domaine complexe et prolifiques de sorte qu'un chercheur, nouveau dans le domaine, puisse rapidement se rapporter aux différentes approches, perspectives et terminologies. Certains taxonomies se concentrent sur les fonctions du geste, d'autres classent les gestes en fonction de leur morphologie tandis que d'autres encore brouillent la distinction entre forme et fonction.

Parmi les nombreuses fonctions du geste, je me concentre sur le rôle des gestes bras-main-doigt comme un moyen d'interagir avec un système informatique. La question conséquente concerne les formes qui prennent en charge ce rôle de manière efficace dans les perspectives à la fois des humains et du système. Pour cela faire, j'utilise la taxonomie de Karam *et al.* comme base. La première contribution de leur travail c'est qu'il est le résultat d'une revue de la littérature de plus de 40 ans d'interaction gestuelle ; en deuxième lieu, il s'agit d'un tentatif de proposer une terminologie unificatrice. J'ai ensuite étendu ou connecté la taxonomie de Karam avec les taxonomies plus spécifiques tels que celle de Cadoz pour les gestes instrumentaux, ou celle de Roudaut et de Baglioni pour les dispositifs mobiles à la main. Le résultat de ma synthèse est représenté dans la Figure 2.6 et le tableaux 2.7 et 2.8 à la fin du chapitre.

Ce chapitre est organisé selon les sections suivantes :

Les fonctions du geste introduit les gestes caractérisée en fonction de leurs caractéristiques des communication.

La morphologie ou style des gestes organise l'état de l'art à partir d'un point de vue morphologique.

Les gestes et leurs technologies habilitantes présente un troisième point de vue qui caractérise les gestes selon les technologies impliquées dans la mise en œuvre de l'acquisition des gestes même et de leurs reconnaissance.

Enfin un point de vue unifié permettra de comparer les approches proposées et de clarifier le vocabulaire adopté.

Une taxonomie pour les techniques d'interaction basées sur les accéléromètres

Dans ce chapitre, je propose une taxonomie permettant de classer les techniques d'interaction sur dispositifs mobiles à base d'accéléromètres. La motivation pour limiter la couverture de la taxonomie aux interactions à base d'accéléromètres est que l'interaction gestuelle pour les appareils mobiles est un domaine très vives et non structurées de recherche. En outre, les accéléromètres sont actuellement la technologie la plus répandue pour la détection de multiples dimensions d'actions dans le monde réel [Hinckley 00]. Le défi est de fournir un cadre de classification qui est à la fois complet et simple à utiliser. Attendu que l'exhaustivité est illusoire dans un domaine dynamique et prolifiques tels que la conception d'interfaces utilisateur, je ne la considère pas comme une de mes objectifs. Je montre, cependant, que la taxonomie est en mesure d'aller au-delà des techniques basées sur des accéléromètres, couvrant un large domaine de questions liées à l'Interaction Homme-Machine.

Pour développer cette taxonomie, j'ai construit un vocabulaire contrôlé (c.à.d. des primitives) obtenue grâce à une analyse approfondie des taxonomies qui ont jeté les bases de l'IHM depuis plus de vingt cinq ans. Souvent, ces premiers travaux en IHM ont été ignorés ou oubliés par les chercheurs entraînés par le mode et par une approche qui poursuit surtout les innovations technologiques.

Ma taxonomie est basée sur les principes suivants :

1. L'interaction entre un système informatique et un être humain est véhiculée par l'entrée (la sortie) des expressions qui sont produites avec les dispositifs d'entrée (de sortie), et qui sont conformes à un langage d'interaction d'entrée (de sortie).

2. Comme touts les langages, un langage d'interaction d'entrée (de sortie) peut être définie formellement en termes de sémantique, syntaxe, et d'unités lexicales.

3. La génération d'une expression d'entrée (de sortie) implique l'utilisation des appareils dont les caractéristiques, du point de vue humain, ont un fort impact sur l'expressivité et l'efficacité de l'interface utilisateur [Buxton 83].

S'appuyant sur les travaux de Foley [Foley 90b] ainsi que sur les considérations pragmatiques des structures d'entrées de Buxton [Buxton 83], ma taxonomie rassemble les quatre aspects de l'interaction allant de la sémantique à la pragmatique avec une extension motivée par l'approche humain pour aborder la spécificité de l'interaction gestuelle basée sur les accéléromètres. Contrairement à la Mackinlay *et al.* et a son analyse sémantique de l'espace de conception pour les dispositifs d'entrée [Mackinlay 90], je ne considère pas les fonctions de transformation qui caractérisent les techniques d'interactions du point de vue système.

Ce chapitre est organisé comme suit : Premièrement, je passe en revue les taxonomies qui ont servi de sources d'inspiration pour mon propre travail : la taxonomie de Foley, pour avoir identifié les tâches génériques de base composant les interactions avec les interfaces graphiques utilisateur ; les taxonomies de Buxton *et al.* et de Card *et al.* pour leurs conclusions sur les dispositifs d'entrée. L'analyse de ces taxonomies offre l'opportunité de clarifier la terminologie (après tout, qu'est ce que c'est une technique d'interaction ?). Ensuite, j'applique l'espace de classification proposé à des techniques d'interaction à la souris bien connues. Dans le chapitre suivant, je présente ma taxonomie illustrée par une étude des techniques d'interaction gestuels à base d'accéléromètres . Je conclus avec les orientations futures pour la recherche que ma taxonomie a permis de découvrir. L'attente est de fournir des nouvelles idées et de proposer des orientations prometteuses pour la conception de nouvelles et puissantes techniques d'interaction gestuelle.

Un état de l'art des techniques gestuelles à base d'accéléromètres

Les techniques interaction basées sur les accéléromètres considérées dans ce chapitre sont présentées en ordre chronologique et tracées dans la Figure 4.3. Pour raison d'exhaustivité, toutes les variations d'une technique d'interaction sont discutées. Par exemple, une technique d'interaction qui existe comme continu (c'est à dire non embrayé) et embrayé apparaît deux fois dans l'espace taxinomique, chacune notée avec les modificateurs syntaxiques appropriés.

Ce chapitre est structuré comme suit : d'abord, j'applique ma taxonomie pour un certain nombre de techniques d'interaction représentatives basées sur les accéléromètres. Ensuite, j'analyse l'image fournie par la taxonomie pour définir le cadre général des interactions existant. La discussion conduira le lecteur à travers l'analyse de l'état de l'art et permettra d'analyser les différentes approches dans le domaine (utilisateur vs. développeur vs. chercheur). La taxonomie proposée permettra de définir la terminologie que je vais utiliser à travers cette recherche doctorale.

Composition de l'interaction tactile et gestuelle

La technique d'interaction TouchOver

Contrairement à la souris, les ècrans tactiles des dispositifs mobiles n'ont pas d'ètat *mouseover* pour fournir à l'utilisateur des informations dynamiques pro-actives. De plus, sur écran tactile, la détection des actions «appuyer» et «relâcher» du doigt rend difficiles les sélections requérant une grande précision.

En réponse à ces limitations, je proposons *TouchOver*, une technique multimodale pour dispositif mobile qui tire partie de l'écran tactile et des accéléromètres : le positionnement est effectué avec le doigt sur la surface tactile et la sélection par inclinaison du dispositif vers l'avant. Ainsi, *TouchOver* introduit un état *mouseover* et améliore la précision de la sélection tout en restant compatible avec les techniques d'interaction existantes. Dans une étude formelle, je compare *TouchOver* à deux autres techniques de sélection. Les résultats montrent une amélioration significative de la précision ainsi qu'un bon compromis entre vitesse d'exécution et précision.

Des menus à base de gestes

Caractérisation des menus gestuels

L'émergence des appareils électroniques a amplifié la nécessité pour la création de périphériques d'entrée de nouvelle génération et des techniques d'interaction pour soutenir un nombre croissant de fonctions et commandes. Il s'agit notamment de la combinaison de commutateurs physiques, des touches avec des fonctions spéciales, des modes et quasimodes logiciels ainsi que des widgets d'usage général comme les boutons logiciels, les barres de défilement, les listes et les menus.

Parmi les widgets actuellement disponibles pour la conception d'interfaces utilisateur, les menus jouent un rôle clé. Ils soutiennent une tâche fondamentale et fréquent de l'homme : celle de faire des choix. Par contraste avec les commandes par terminaux et clavier, les menus présentent aux utilisateurs les choix possibles (soit graphiquement ou vocalement), et seuls les choix qui sont sémantiquement valides dans l'état actuel du système. *À travers les menus, toutes les actions possibles peuvent être rendue visible et, par conséquent, facilement détectable* [Norman 10]. Ils sont une *alternative intéressante* [à la saisie de commandes par clavier], car ils peuvent éliminer l'apprentissage et la mémorisation des séquences de commandes complexes [Shneiderman 87]. Lorsqu'ils sont conçus avec soin, les menus raccourcissent l'apprentissage, fournissent une structure claire pour la prise de décision, soutiennent l'exploration, réduisent les erreurs, et peuvent être intéressants aussi pour les utilisateurs experts quand ils comprennent des raccourcis [Shneiderman 87].

En raison de leur rôle clé dans la conception des interfaces utilisateur, les menus ont été étudiés depuis les années quatre-vingt. Comme la Figure 6.1 illustre, ils ont évolué comme la technologie a continué d'apporter des contraintes supplémentaires telles que les petits écrans, mais aussi de nouvelles possibilités telles que l'utilisation du geste et de grandes surfaces interactives. Dans ce chapitre, je m'occupe des menus graphiques gestuelle pour les appareils portables à la main. Avant d'entrer dans l'étude détaillée des menus gestuelle, je me propose de répondre à la question «Qu'est-ce qu'un menu ?".

GeLATI : intégration de menus hiérarchiques gestuels dans une boîte à outils

Ce chapitre présente GeLATI, une bibliothèque logicielle permettant la réalisation de menus gestuels. En entrée, elle décompose les gestes en successions de segments rectilignes, et permet leur reconnaissance en les comparant au fur et à mesure de leur exécution à des patrons paramétrables par des fichiers de données. En sortie, le retour graphique offert à l'utilisateur se base sur les patrons d'entrée, mais les détails de son aspect sont eux aussi paramétrables. Je discutes enfin l'intégration de GeLATI dans une boîte à outils de construction d'interface préexistante (celle de l'iOS d'Apple : UIKit) et je montre qu'elle permet d'augmenter des interacteurs classiques.

Conclusions

Conclusions

Les chapitres précédents ont montré l'importance, la diversité et la complexité des techniques d'interaction basées sur le geste pour les dispositifs mobiles. Bien que le geste soit considéré comme une façon naturelle d'interagir avec une machine, le développement de techniques d'interaction gestuelle efficaces reste toujours une tâche difficile. Les concepteurs ne peuvent pas encore compter sur des conventions standard tels que les schémas d'interaction développés pour les interfaces utilisateur WIMP. Au lieu de cela, ils doivent faire face à une poussée technologique perpétuelle qui élargit en permanence l'espace de conception. En outre, la mise en œuvre de systèmes de reconnaissance gestuelle, et de leur intégration dans des boîtes à outils d'interface utilisateur actuelle, n'est pas une affaire simple.

Dans ce contexte, cette thèse se concentre sur les techniques d'interaction gestuelle pour une classe spécifique de périphériques (c.à.d. ordinateurs de poche dotés d'écran tactile et accéléromètres) avec deux principales préoccupations : (1) fournir aux concepteurs un cadre conceptuel qui permet de structurer le raisonnement sur la nature du geste et sur les techniques d'interaction gestuelles, et (2) fournir aux développeurs un algorithme de reconnaissance de gestes efficaces et robustes qui peut être facilement intégrée dans des boîtes à outils existantes.

Contributions

La contribution de cette thèse est triple : conceptuelle, avec des taxonomies pour raisonner sur les gestes et les gestes pour les appareils mobiles qui tiennent dans la main ; techniques, avec la bibliothèque GeLATI ; et interactionnelle, avec la conception de nouvelles technique d'interaction tels que TouchOver.

Classification Gestuelle

Je propose une synthèse des taxonomies qui ont été proposées pour les gestes en psycholinguistique ainsi qu'en l'Interaction Homme-Machine. Certaines d'entre elles se concentrent sur les fonctions du geste même, d'autres classent les gestes en fonction de leur morphologie tandis que d'autres masquent la distinction entre forme et fonction. En corrélant ces perspectives au sein d'un cadre unique, les chercheurs nouveaux venus dans le domaine de l'interaction gestuelle ont une vue unifiée et concise pour rapidement se rapporter aux différentes approches, perspectives et terminologies.

En outre, j'utilise ce cadre pour préciser la couverture de mon travail et pour définir mon propre intérêt : celui de gestes déictiques et sémaphoriques pour l'interaction homme-machine. En ce qui concerne le pouvoir expressif, je m'adresse à des gestes qui couvrent la specification, en une seule action, de commandes et leurs paramètres.

Une taxonomie synthétisée des gestes est appropriée pour comprendre l'ampleur du domaine, mais pas pour le raisonnement à un grain fin sur la conception d'une technique d'interaction gestuelle. À cette fin, je propose une nouvelle taxonomie pour les techniques d'interaction gestuelles basées sur les accéléromètres. Cette taxonomie, motivée par les contributions de base en IHM, rassemble les tâches génériques de Foley avec les dimensions lexicale, syntactique, sémantique et pragmatique des langages afin de caractériser les actions physiques impliquées dans l'interaction gestuelle. Elle est radicalement centrée sur les actions physiques humaines. L'hypothèse est que l'action physique est le niveau atomique approprié à partir duquel de nouvelles techniques d'interaction peuvent être conçues pour fournir un ensemble cohérent de langages au sein d'un système. Dans cette taxonomie, l'abscisse définit le lexique en termes de manipulations physiques que les utilisateurs doivent effectuer, avec une séparation claire entre l'interaction d'arrière et de premier plan. L'ordonnée correspond aux tâches de Foley. Une technique d'interaction est tracée comme un point dans cet espace où chaque point est décoré avec des propriétés pragmatiques et syntaxiques. La taxonomie comprend deux modificateurs syntaxiques : si la technique d'interaction est embrayée ou poursuivie, et le type de contrôle (en position, vitesse ou accélération). Un modificateur pragmatique indique le degré d'indirection de la technique d'interaction.

Afin de démontrer sa flexibilité et sa couverture, cette taxonomie a été appliquée à la classification des techniques d'interaction post-WIMP utilisant les accéléromètres ainsi que pour des techniques d'interaction WIMP.

Parmi les techniques d'interaction existantes, les menus jouent un rôle prédominant. Pour cette raison, j'ai revu la conception des espaces développés pour les menus ainsi que les définitions et les distinctions terminologiques subtiles entre widget, interacteurs, et les instruments d'interaction. A partir de là, je propose une distinction claire entre les instruments logiciels et les instruments matériels ainsi qu'une version révisée de la définition que Bailly donne des menus.

Un geste est un instrument physique dont le composant matériel est un périphérique d'entrée directe (souris, stylo ou écran tactile) et le composant logiciel, un système de reconnaissance gestuelle.

Un menu est un instrument logiciel qui répond aux exigences suivantes :

1. Il prend en charge la sélection d'un élément dans un ensemble fini d'options.
2. Il minimise les efforts mentaux, en présentant l'ensemble des options à l'utilisateur.
3. Il présente les options comme une structure sémantiquement et spatialement significative.
4. Il est transitoire en ce sens qu'il est perceptible à l'utilisateur que lors de son interaction avec l'utilisateur.
5. Il est quasimodal car il définit un contexte local pour l'interprétation du système qui est maintenu explicitement par l'utilisateur jusqu'à ce que l'interaction se termine.

.

Un menu gestuel est un instrument logiciel couplé à un instrument geste tel que le processus de sélection est dirigé par le geste.

GeLATI

GeLATI est une bibliothèque pour concevoir, prototyper et mettre en œuvre les menus gestuels.

Les principales caractéristiques de GeLATI sont les suivantes :

1. La reconnaissance de gestes est basée sur une nouvelle approche vectorielle temps réel, qui combine de multiples entrées pour permettre des interactions inter-modales et/ou multi-modales ;
2. La reconnaissance de gestes, qui utilise une approche à modèle («template») unique, n'a pas besoin d'apprentissage ;
3. Les gestes, qui sont modélisés comme une suite de vecteurs, sont rectilignes. (Les gestes rectilignes ont été démontrés comme plus utilisables que les gestes de forme libre.) ;
4. Les gestes déictiques et sémaphoriques sont pris en charge ;
5. Les utilisateurs débutants sont guidés le long de chemins bien définis pour compléter le geste. Les utilisateurs expérimentés peuvent compléter les commandes sans retour graphique, comme illustré par les Marking Menus ;
6. L'exploration est soutenue par un mécanisme incrémental qui permet le retour au niveau supérieur de la hiérarchie du trait courant ;
7. Les menus hiérarchiques gestuels peuvent être automatiquement intégrés dans les outils graphiques existantes.

En synthèse, le reconnaisseur gestuel GeLATI est capable d'offrir les mêmes caractéristiques que les approches basées statistiques, mais sans l'infrastructure statistique accélérant ainsi le processus de reconnaissance. De la même façon que $1, la reconnaissance est résistante aux variations de l'échantillonnage. Par contraste avec $1, l'objectif est de soutenir la variance de rotation qui est une caractéristique importante des gestes humains. Si nécessaire, la variance en échelle et en position (spécifiable par le développeur) est également supportée. Enfin, la reconnaissance est en mesure de proposer dynamiquement les gestes candidats lors de l'interaction elle-même non pas seulement lorsque les utilisateurs ont terminé leur geste.

Un certain nombre d'exemples ont été mis en œuvre pour démontrer la couverture fonctionnelle de GeLATI ainsi que son intégration dans les outils graphiques.

TouchOver

TouchOver est une technique d'interaction multimodale complémentaire pour les interactions à une main avec les appareils de poche dotés d'écran tactile et d'accéléromètres.

TouchOver offre un modèle d'entrée à trois états, similaire à celui déjà connu dans les interfaces au stylet, où le système trace les mouvements du doigt, ajoutant ainsi un état de suivi passif à l'entrée tactile. Cela crée de nouvelles opportunités pour de nouvelles techniques d'interaction, pour dispositifs portables, comme le survol, le feedforward, ou les interactions qui ne nécessitent pas l'attention visuelle.

TouchOver a été validé expérimentalement en précision et vitesse.

Limitation et perspectives

Deux principales limites doivent être abordées dans le cadre de GeLATI : (1) le mécanisme de re-sélection et (2) l'intégration de gestes non rectilignes.

Le mécanisme de re-sélection

Lorsqu'un utilisateur interagit avec un menu GeLATI, il/elle traverse plusieurs noeuds en suivant simplement les chemins visibles. Lors de la correction de son choix, l'utilisateur remonte le chemin d'interaction. Dans un menu traditionnel l'état interne est caractérisé par la position du curseur dans les objets graphiques le composant. Dans un menu GeLATI, la position de l'utilisateur est vérifiée par rapport au dernièr point dans chaque trait qui compose le geste. Une analyse intéressante pourrait être de tester la satisfaction de l'utilisateur et de valider le nouvelle approche en vue d'intégrer l'algorithme dans d'autres reconnaisseurs gestuels existants. Une autre façon est d'introduire le concept de boîte englobante dans GeLATI, fusionnant ainsi les approches dirigées par la position ou par les gestes.

Traits non rectilignes

L'algorithme GeLATI est basé sur l'hypothèse principale qu'une séquence de traits linéaires (vecteurs) compose des formes complexes guidant ainsi les utilisateurs au cours de l'interaction. Un scénario intéressant qui devrait être évalué consiste à étendre les modules de base qui composent les gestes complexes aux traits non-linéaires. Par exemple, un geste pourrait être défini comme une séquence de $y = sin(x)$ plus un trait de $y = x$ plutôt que la simple vectorialisation proposée.

J'aurais dû...

L'applicabilité de l'approche GeLATI a été testée dans plusieurs expérimentations informelles et dans différents démonstrateurs. Néanmoins, l'architecture proposée et la bibliothèque logicielle ont encore besoin d'être testées avec d'autres programmeurs et concepteurs. En outre une étude formelle utilisateur doit être conçue afin de vérifier l'acceptation des utilisateurs à intégrer les gestes dans les widgets classiques.

Appendix A.

Technical Annex

A.1. API

An overview of the GeLATI class hierarchy

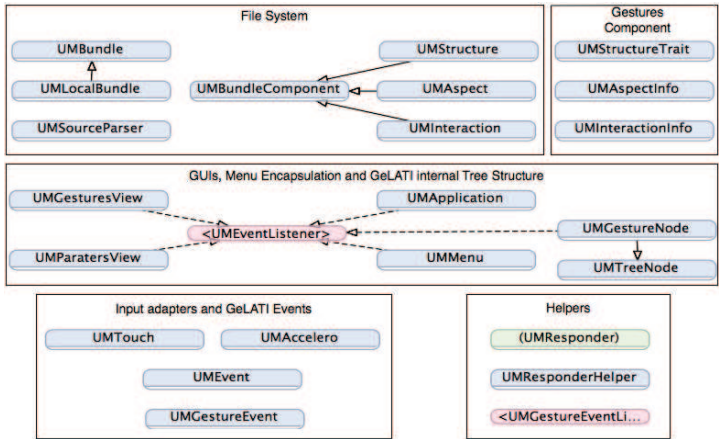


Figure A.1.: Five main blocks compose the GeLATI library. **File System** contains those classes (the Bundle) needed to load specification files from device memory. **Gestures Components** contains those classes representing the three main gesture components Interaction, Aspect and Structure. **GUIs, Menu Encapsulation and GeLATI Internal Tree Structure** contains all classes interested in raw events used to control and update the menu status. **Input Adapters and GeLATI Events** capture input event from input devices (Touch Screen and Accelerometers) and transform them into GeLATI events. GeLATI core generate the gesture event sent to the application. **Helpers** contains the glue classes needed to integrate in existing UIKit class hierarchy and to build compatible application.

A.1.1. UMMenu header

```
1 #import <UIKit/UIKit.h>
2 #import "UMProtocols.h"
3 @class UMBundle;
4
5 /**
6  * This is the main interface a UMMenu will implement.
7  * It contains the basic operations in order to add/remove a
        bundle to/from the menu
8  * and to add/remove listeners
9  */
10 @interface UMMenu : NSObject <UMEventListener> {
11     /* The objects that will receive the gestures actions, i.e.
            that will listen for commands*/
12     NSMutableArray *_delegates;
13     /* The objects that will receive raw events after being
            traited by the menu */
14     NSMutableSet *_listeners;
15     /* The bundle loaded by the current menu */
16     UMBundle *_bundle;
17
18     /* Reusable pool to optimize trees creation/deletion */
19     NSMutableSet *_treesPool;
20
21     /* The current active trees */
22     NSMutableDictionary *_activeInteractions;
23     /* The current user active interactions*/
24     NSMutableDictionary *_gesturalInteractions;
25     /* Current candidate gestures*/
26     NSMutableDictionary *_gesturesSet;
27
28     /* For Future Use in multifingers interactions  */
29     NSMutableDictionary *_mergedGestures;
30 }
31
```

```
32 @property(readonly) NSDictionary *activeInteractions;
33 @property(readonly) NSDictionary *gesturalInteractions;
34 @property(readwrite,retain) UMBundle *bundle;
35
36 /**
37  * Register and load the Gesture Bundle
38  */
39 - (void)setBundleNamed:(NSString *)bundleName;
40 /**
41  * Add a UMGestureEvent Delegate
42  */
43 -(void)addDelegate:(id)output;
44 /**
45  * Remove a UMGestureEvent Delegate
46  */
47 -(void)removeDelegate:(id)output;
48 /**
49  * Add a UMEvent Listener
50  */
51 - (void)addEventListener:(id<UMEventListener>)aListener;
52 /**
53  * Remove a UMEvent Listener
54  */
55 - (void)removeEventListener:(id<UMEventListener>)aListener;
56 /**
57  * Used by graphical interfaces to retrive the tree root node
        given an interaction ID
58  */
59 -(UMGestureNode *)treeForInteraction:(UMEvent *)event;
```

A.1.2. QuestionMark View Controller

```
1 #import "Gelati2.h"
2
3 @implementation ViewController
4
```

```
5  - (void)viewDidLoad {
6      [super viewDidLoad];
7
8      /* Create a new UMMenu */
9      UMMenu *myMenu = [[UMMenu alloc] init];
10     /* Add a gesture set to the UMMenu */
11     [myMenu setBundleNamed:@"questionmark.umenu"];
12     /* Register the Gestural Interaction to the desired
           graphical view */
13     [self.view addGelatiMenu:myMenu];
14
15     /* Register a graphical interfaces */
16     [aMenu addEventListener:self.parametersView];
17     /* Register a graphical interfaces */
18     [aMenu addEventListener:self.gesturesView];
19     /* Register a graphical interfaces */
20     [aMenu addEventListener:self.boxView];
21
22     /* Register a gesture event delegate */
23     [myMenu addDelegate:self];
24 }
25
26 #pragma mark -
27 #pragma mark Menu delegate
28
29 - (void)help:(UMGestureEvent *)event
30 {
31     /* A gesture event method example */
32
33     [event retain];
34     switch (event.state) {
35         case UMGestureBegan:
36             break;
37         case UMGestureContinued:
38             break;
39         case UMGestureCancelled:
```

```
40                  break;
41              case UMGestureEnded:
42                  [self doTheWork];
43                  break;
44              default:
45                  break;
46          }
47          [event release];
48      }
49
50  @end
```

A.2. XML Files

A.2.1. QuestionMark GeLATI Definition: Structure.plist

```
1  {
2    deadZoneLength = 25;
3    "polarZoneUpperPrecision" = "0.3";
4    "polarZoneLowerPrecision" = "0.4";
5    polarZoneLenght = 90;
6    linearZoneEnabled = YES;
7    slidingWindowLength = 120;
8    "minimunDistanceInTrait" = 70;
9    proximityThreshold = 20;
10   items = (
11     {
12       id = Vector;
13       direction = "0.78";
14     },
15     {
16       id = "Question Mark";
17       direction = "3.93";
18       slidingWindowLength = 150;
19       items = (
20         {
```

```
21            id = "Question Mark";
22            direction = "5.50";
23            items = (
24              {
25                id = "Question Mark";
26                direction = "0.78";
27                items = (
28                  {
29                    id = "Question Mark";
30                    direction = "2.36";
31                    items = (
32                      {
33                        id = "Question Mark";
34                        direction = "1.57";
35                        "polarZoneUpperPrecision" = "0.4";
36                        "polarZoneLowerPrecision" = "0.3";
37                      },
38                    );
39                  },
40                );
41              },
42            );
43          },
44        );
45        options = {
46          roundEdges = YES;
47        };
48      },
49    );
50  }
```

A.2.2. QuestionMark GeLATI Definition: Interaction.plist

```
1  {
2    EnableTiming = NO;
3    timer = 10;
```

```
4   rootSelector = "touch:";
5   defaultSelector = "incompleteGesture:";
6   completeGesture = NO;
7   continuous = NO;
8   detachWhenCompleted = NO;
9   speedLevels = (
10    "0.0",
11    "0.18",
12    "0.55",
13  );
14  items = (
15    {
16      id = Vector;
17      completeGesture = YES;
18      selector = "help:";
19    },
20    {
21      id = "Question Mark";
22      items = (
23        {
24          items = (
25            {
26              items = (
27                {
28                  items = (
29                    {
30                      completeGesture = YES;
31                      selector = "help:";
32                    },
33                  );
34                },
35              );
36            },
37          );
38        },
39      );
```

```
40      },
41    );
42  }
```

A.2.3. QuestionMark GeLATI Definition: Aspect.plist

```
1  {
2    unitType = pixel;
3    unit = 2;
4    "maximumComponentToDraw" = 10;
5    "constantSectionLenghtToDraw" = 15;
6    backgroundImage = "resources/spot.png";
7    cursorImage = "resources/spot.png";
8    rootID = Root;
9    colorName = white;
10   items = (
11     {
12       id = Vector;
13       color = (
14         "0.4",
15         "0.2",
16         "0.4",
17         1,
18       );
19     },
20     {
21       id = "Question Mark";
22       color = (
23         "0.3",
24         "0.3",
25         "0.4",
26         1,
27       );
28       items = (
29         {
30           id = "Question Mark";
```

```
31            color = (
32              "0.3",
33              "0.3",
34              "0.4",
35              1,
36            );
37            items = (
38              {
39                id = "Question Mark";
40                color = (
41                  "0.3",
42                  "0.3",
43                  "0.4",
44                  1,
45                );
46                items = (
47                  {
48                    id = "Question Mark";
49                    color = (
50                      "0.3",
51                      "0.3",
52                      "0.4",
53                      1,
54                    );
55                    items = (
56                      {
57                        id = "Question Mark";
58                        color = (
59                          "0.3",
60                          "0.3",
61                          "0.4",
62                          1,
63                        );
64                      },
65                    );
66                  },
```

```
67                  );
68                },
69              );
70            },
71          );
72          options = {
73            roundEdges = YES;
74          };
75        },
76      );
77  }
```

Colophon

This thesis was made in LaTeX using the "hepthesis" class [Buckley 11].

List of Figures

1.1. From the left: a Microsoft Windows Mobile 7 interface screenshot showing a perspective animation; the Apple iPhone home screen; the Palm Pre task selection interface. 6

1.2. Krueger is one of the pioneer researchers of full body gesturing. 8

1.3. Charade uses a 3D hand gestures interaction model. On the right, an example of the notation used to denote the "next chapter" gesture [Baudel 95]. 8

1.4. Villamore *et al.* propose a reference guide for all touch-based gestures implemented in current modern systems such as iOS, Windows Mobile 7 or WebOS-based mobile devices [Villamore 10]. 9

1.5. OctoPocus integrates feedback with feedforward in gesture-driven interfaces to help users to discover and learn gestural menus [Bau 10]. 9

1.6. Roudaut proposes the exploitation of micro-gestures on touch screen based devices with “Microrolls” [Roudaut 10]. 10

1.7. TimeTilt proposes an example of multimodal interaction that exploits both accelerometers coupled with two different interaction languages [Roudaut 10]. 11

1.8. Bolt used the combination of voice and hand gestures to command the “Media Room” to controll shape based application [Bolt 80]. 11

2.1. Instrumental Interaction provide Humans with a means to interact with the Environment through the Machine [Cadoz 94]. 24

2.2. Gesture in human computer interaction [Cadoz 94]. Instrumental gesture is primarily ergotic (it produces energy). It may be used for communication (it is then semiotic) or for pure action on the world (absence of meaning) [Cadoz 94]. 25

2.3. The gestural interaction framework proposed by Karam *et al.* [Karam 05] enables a complete discussion about gestures themselves and the contextual variables/parameters associated with. 27

2.4. Baglioni's characterization space enhance Karam's taxonomy to adapt to handheld devices. Adapted from [Baglioni 09]. 29

2.5. Hinckley *et al.* propose a complete design space of motion sensing interaction techniques [Hinckley 11]. Highlithed the section of the design space this thesis focuses on. 33

2.6. A synthetic representation of the gestures terminology used in different taxonomies. 34

2.7. An integrated view of the analyzed taxonomies highlights the relationshinps among different approaches and metaphors. 36

2.8. Correlation between gesture formality and expressive power. 37

3.1. Ballagas taxonomy plots existing interaction techniques implemented by using the phone as input device [Ballagas 06]. 43

3.2. Physical, virtual and composite input devices classified within Mackinlay's *et al.* taxonomy. A circle in a cell indicates that a device senses a physical property characterized by the coordinates of the grid. A black line represents a merge composition. An arrow represents a connect composition. A dashed line - no example shown here - a layout composition [Mackinlay 90]. 46

3.3. The "sliding" gesture is semantically multiplexed to achieve different meanings, depending on context. 46

3.4. A new classification space for gestural interaction techniques based on accelerometers. The abscissa defines the lexicon in terms of the physical manipulations users perform with the device, with a clear separation between background and foreground interaction. The ordinate corresponds to Foley's interaction tasks. An interaction technique is uniquely identified by an integer i and plotted as a point in this space. Each point is decorated with the pragmatic and syntactic properties of the corresponding interaction technique. There are two syntactic modifiers: an oval indicates whether the interaction technique is clutched or continued, and an exponent expresses the control type (position, speed, or acceleration). F, which is the only pragmatic modifier, indicates the degree of indirection of the interaction technique. 47

3.5. Classical mouse-driven interaction techniques within my taxonomy: (1) Positioning the cursor; (2) Menu item selection; (3) Defining orientation in a graphics editor; (4) Sketching using a drawing tool; (5) Defining a quantity through a slider; (6) Typing text with a virtual keyboard. . . . 51

4.1. Fitzmaurice et al. used the *Move* action to command the history of their spreadsheet, while the *Tilt* of the device let the user span the cells around the selected [Fitzmaurice 93]. 56

4.2. Rekimoto's tilt based menu control the navigation in a map [Rekimoto 96]. 57

4.3. A state of the art of accelerometers-based interaction techniques. An interaction technique is identified by an integer i: (7) successive undo/redo as well as active cell selection through translations; (8) tilt to preview adjacent cells; (9) tilt to select a command in a pie menu; (10) tilt to select commands in linear and pie menus; (11) tilt to control position on a map; (12) tilt to browse a calendar; (13) squeeze to stop an interaction; (14) drawing through physical translations; (15) passive screen orientation adaptation; (16) active screen orientation control; (17) tilt to select pictures; (18) tilt to control first person shooter game; (19) tilt to enter text; (20) passive control of screen orientation and power energy saving; (21) tilt and translation to select physical world object; (22) control volume through tilt; (23) translation of virtual workspace through physical translation; (24) selection of the level of user interface details through translation; (25) gestural authentication with shock durations over time; (26) shake to quantify device status; (27) tilt to select graphical views; (28) shock to trigger an interaction; (29) shock to select the previously active application; (30) gesture recognition; (31) shake to select the next song; (32) tilt to quantify the zoom factor; (33) tilt to control screen/contextual objects rotation; (34) shock the device to select a command: delete current picture or undo deletion; (35) tilt de device to select the crop command; (36) passive control of touch force to select the user desired action; (37) passive control of touch force to select dragging command. 58

4.4. Partridge et al. propose to map a position-based interaction technique drive by accelerometers to a menu containing the most used characters [Partridge 02]. 60

4.5. Tapping the back of the device trigger the continuous/discrete mode. In continuous mode application switch is activated by tilting the device. In discrete mode application switch is achieved using *jerk movements* [Roudaut 09]. 62

5.1. With touchOver, users can switch between two interaction states (a) and (b) with a simple tilt of the device while still interacting with their finger. This permits for example interactions such as hovering, dragging, feedforward enabled techniques, visual and eye-free interface exploration, and selection precision improvement. 72

5.2. Mouse input state machine (reproduced from [Buxton 90]). 74

5.3. Stylus on graphics tablet input state machine (reproduced from [Buxton 90]). 74

5.4. Touch-screen input state machine (reproduced from [Buxton 90]). 74

5.5. The TouchOver input state machine. 77

5.6. The user interface for the precision experiment: the target line (left); the thumb approaching the target (center); the thumb is on the target (right). 81

5.7. Speed experiment application interface: left the first target; right the second target. 81

5.8. Percentage of errors by validation techniques 83

5.9. Boxplot of error distance in millimeters grouped by validation technique . 83

5.10. Boxplot of validation duration for *Take-Off* and *TouchOver* 85

5.11. Comfort and efficiency grades for each technique for the precision experiment 86

5.12. Comfort and efficiency grades for each technique for the speed experiment 86

6.1. Menus declinations. From left: the Wavelet menu for touch-enabled mobile devices [Francone 09]; Polymorphic Menu developed in the NOMAD project [http://iihm.imag.fr/contract/nomad/], MTM (Menu MultiTouch) for multipoints tabletop [Bailly 09]; Shadow Reaching : target selection using shadows [Shoemaker 07]. 91

6.2. Different menu structures impose different approaches and learning difficulties on users [Shneiderman 87]. 101

6.3. Shneiderman's hierarchical menus in a modern WIMP environment. . . . 103

6.4. Shneiderman's binary menus in a moder WIMP environment. 104

6.5. Bailly's MenUA classification space [Bailly 09]. 104

6.6. Marking menus propose a Novice and an Expert mode. On the left, a marking menu used in novice mode resembles a Pie Menu [Callahan 88] but user's movements are tracked leaving an ink trail. On the right, the marking menu when used in expert mode: the user makes a mark without relying on the graphical representation [Kurtenbach 93]. 107

6.7. A hierarchical marking menus (left) enriched with feedforward (right) that permits the user to foresee each command's hierarchical content [Bau 08]. 109

6.8. StrokeShortcuts integrated in a media player. On the left, gesture shortcuts are visible while using a classical linear menu. On the right, the "open playlist" corresponding gestures. Help fades away on a timeout. 110

6.9. Ideal Flower menus support up to 56 items per hierarchy level (left). Flower menus can control the most used commands in a simple hierarchy level (right) [Bailly 09]. 111

6.10. Leaf menu for touch-screen enabled mobile devices where curvature is exploited to discriminate gestures [Bailly 09]. 113

7.1. A set of 13 features characterizing each gesture in GRANDMA statistical gestures recognizer [Rubine 91]. 121

7.2. $1 gesture recognizer first two steps. At the top, a raw gesture as captured by the input device with three different resamples where N denotes the number of sampling points. Wobbrock *et al.* have determined empirically to use $N = 64$ as a reference resampling parameter. At the bottom of the figure, the second step of the gesture recognizer. The resampled path is rotated to an "indicative angle" to ease the recognition and the match process. [Wobbrock 07]. 123

7.3. A vectorial approach to gesture recognition. From left to right, the "vectorialization" of a gestural template. 125

7.4. On the left, a simple gesture composed of a single trait: for the simplest cases, an angle (α) is the only parameter needed (direction) to define a gesture of infinite length (the vector magnitude). On the right, a screenshot of a vector with all the GeLATI's parameters represented graphically. . . 126

7.5. On the left, a GeLATI "Question Mark" as a series/concatenation of five vectors. In the center, a screenshot of the question mark gesture along with the feedback provided by GeLATI. On the right, a screenshot with a different feedback that shows the name of the command (help) as well as the exact trace of the user input. 127

7.6. GeLATI is implemented in Objective-C on top of the iOS Foundation Framework. Default GeLATI GUIs are developed over the GeLATI Core using the Core Graphics and Quartz frameworks. A GeLATI menu is built over the GeLATI core and organizes multiple gestures through an entry point hierarchical root. Applications built on top of GeLATI receive the method specified by in gesture definition. 129

7.7. Raw events are transformed into UMEvents and passed to the GeLATI core. GeLATI consumes the events by updating the traits that compose the menu tree. UMEvents are passed to the feedback/feedforward views together with the tree in order to update the aspect graphical rendering. UMGestureEvents are generated and sent to the application to update its status and/or to fire the developer's defined actions. 131

7.8. Different GUI aspects can be associated to the same gestural menu. The first raw shows the two default GUI aspects integrated in the library. The first one is an accurate representation of the functioning of the underlying algorithm. The second one is an OctoPocus-like view of a GeLaTI menu. The second row shows a third GUI aspect designed to imitate the Marking Menu. 133

7.9. Three gestures are associated to the same vector with three different lengths. An example cannot be supported by the $1. 134

7.10. The "Branch" gesture has a fixed length and two siblings: "Branch Right" and "Branch Left". 134

7.11. GeLATI gestural menus can be integrated into legacy widgets. Enhancing classical GUI components with gestural interaction permits a complete gestural experience and optimized screen space management. 135

7.12. Two screenshots showing the test application and the interaction I designed to test the accelerometers+touch screen synaesthesia. In 7.12(a) the application is waiting for a new interaction to start. In 7.12(b) the interaction started and continued selecting the cheese on the left through a finger movement. The task of the active trial is to select the yellow upper-left cheese, while the currently selected cheese is shown in green. User could either control the application through the touch screen or though the accelerometers. 136

7.13. Multiple instances of a GeLATI menu can be activated with the same GUI aspect or with different ones. On the left two GeLATI menus have been activated on the main view. On the right two menus have been activated on the same graphical button. The "BRigth" options controls the button orientation while "Left" option control the widget position. 138

A.1. Five main blocks compose the GeLATI library. **File System** contains those classes (the Bundle) needed to load specification files from device memory. **Gestures Components** contains those classes representing the three main gesture components Interaction, Aspect and Structure. **GUIs, Menu Encapsulation and GeLATI Internal Tree Structure** contains all classes interested in raw events used to control and update the menu status. **Input Adapters and GeLATI Events** capture input event from input devices (Touch Screen and Accelerometers) and transform them into GeLATI events. GeLATI core generate the gesture event sent to the application. **Helpers** contains the glue classes needed to integrate in existing UIKit class hierarchy and to build compatible application. . . 195

List of Tables

2.1. Roudaut's Gestural interaction language applied to mobile devices platforms [Roudaut 10]. 32

3.1. Foley's classification of fundamental interaction tasks expresses the requirements that interaction techniques must satisfy [Foley 90b]. 41

4.1. Hinckley classification for Tilt/Accelerometer input devices [Hinckley 00]. 61

6.1. Shneiderman's menu selection guidelines distilled from practice [Shneiderman 87].103

6.2. Roudaut classifies gestural menus according the gestures type they are driven by and the layout menu items are disposed on [Roudaut 10]. . . . 114

6.3. The different layouts gestural menus are characterized by, imply consequences on the interaction they propose. Here a synthesis of the pros and cons proposed by Roudaut [Roudaut 10]. 115

Bibliography

[Accot 97] Johnny Accot & Shumin Zhai. *Beyond Fitts' law: models for trajectory-based HCI tasks*. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '97, pages 295–302, New York, NY, USA, 1997. ACM.

[Albinsson 03] Par-Anders Albinsson & Shumin Zhai. *High precision touch screen interaction*. Proceedings of the conference on Human factors in computing systems - CHI '03, no. 5, page 105, 2003.

[Appert 04] Caroline Appert, Michel Beaudouin-Lafon & Wendy E. Mackay. *Context matters: Evaluating Techniques with the CIS Model*. People and Computers XVIII - Design for Life, HCI 2004, 2004.

[Appert 09] Caroline Appert & Shumin Zhai. *Using strokes as command shortcuts: cognitive benefits and toolkit support*. In Proceedings of the 27th international conference on Human factors in computing systems, pages 2289–2298. ACM, 2009.

[Arc 92] *A Metamodel For The Runtim E Architecture Of An Interactive System*. SIGCHI Bulletin, vol. 24, no. 1, pages 32–37, 1992.

[Axtell 91] R.E. Axtell. Gestures: The do's and taboos of body language around the world. Wiley, 1991.

[Baglioni 09] Mathias Baglioni, Eric Lecolinet & Yves Guiard. *Espace de caractérisation des interactions gestuelles physiques sur dispositifs mobiles*. Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine - IHM '09, page 203, 2009.

[Bailly 09] Gilles Bailly. *Techniques de menus: Caractérisation, Conception et Evaluation*. PhD thesis, Université Joseph Fourier, 2009.

[Ballagas 06] Rafael Ballagas, Jan Borchers, Michael Rohs & Jennifer G. Sheridan. *The Smart Phone: A Ubiquitous Input Device.* IEEE Pervasive Computing, vol. 5, no. 1, pages 70–77, January 2006.

[Bartlett 02] Joel F. Bartlett. *Rock'n'Scroll is here to stay.* Computer Graphics and Applications, IEEE, vol. 20, no. 3, pages 40–45, 2002.

[Bau 08] Olivier Bau & Wendy E. Mackay. *OctoPocus: a dynamic guide for learning gesture-based command sets.* In Proceedings of the 21st annual ACM symposium on User interface software and technology, pages 37–46. ACM, 2008.

[Bau 10] Olivier Bau. *Interaction streams.* PhD thesis, INRIA Saclay/LRI, Orsay, 2010.

[Baudel 93] Thomas Baudel & Michel Beaudouin-Lafon. *Charade: remote control of objects using free-hand gestures.* Communications of the ACM, vol. 36, no. 7, pages 28–35, 1993.

[Baudel 95] Thomas Baudel. *Aspects Morphologiques de l'Interaction Humain-Orinateur: Étude de Modèles d'Interaction Gestuels.* PhD thesis, Université Paris XI Orsay, 1995.

[Beaudouin-Lafon 00] Michel Beaudouin-Lafon. *Instrumental interaction: an interaction model for designing post-WIMP user interfaces.* In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 446–453. ACM, 2000.

[Benko 06] Hrvoje Benko, Andrew D. Wilson & Patrick Baudisch. *Precise selection techniques for multi-touch screens.* Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06, page 1263, 2006.

[Benko 08] Hrvoje Benko, Andrew D Wilson & Ravin Balakrishnan. *Sphere: multi-touch interactions on a spherical display.* In Proceedings of the 21st annual ACM symposium on User interface software and technology, UIST '08, pages 77–86, New York, NY, USA, 2008. ACM.

[Blanch 04] Renaud Blanch, Yves Guiard & Michel Beaudouin-Lafon. *Semantic pointing: improving target acquisition with control-display ratio*

adaptation. Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '04, vol. 6, no. 1, pages 519–526, 2004.

[Bolt 80] Richard A. Bolt. *"Put-that-there": Voice and gesture at the graphics interface*. In Proceedings of the 7th annual conference on Computer graphics and interactive techniques, pages 262–270. ACM, 1980.

[Bragdon 11] Andrew Bragdon, Eugene Nelson, Yang Li & Ken Hinckley. *Experimental Analysis of Touch-Screen Gesture Designs in Mobile Environments*. In Proceedings of the 2011 annual conference on Human factors in computing systems, pages 403–412, 2011.

[Brooks Jr 88] Frederik P. Brooks Jr. *Grasping reality through illusion—interactive graphics serving science*. Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88, pages 1–11, 1988.

[Buckley 11] Andy Buckley. *The hepthesis {\LaTeX} class*, 2011.

[Buxton 83] Bill Buxton. *Lexical and pragmatic considerations of input structures*. ACM SIGGRAPH Computer Graphics, vol. 17, no. 1, pages 31–37, January 1983.

[Buxton 90] Bill Buxton. *A three-state model of graphical input*. In Human-computer interaction-INTERACT, volume 90, pages 449–456. Citeseer, 1990.

[Buxton 95] Bill Buxton. *Integrating the periphery and context: A new taxonomy of telematics*. In Proceedings of graphics interface, volume 95, pages 239–246, 1995.

[Cadoz 94] Claude Cadoz. *Le geste canal de communication homme/machine*. Technique et science informatiques, page 31, 1994.

[Callahan 88] J Callahan, D Hopkins, M Weiser & B Shneiderman. *An empirical comparison of pie vs. linear menus*. In CHI'88: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 95–100. ACM Press, 1988.

[Card 91] Stuart K. Card, Jock Mackinlay & George G. Robertson. *A*

morphological analysis of the design space of input devices. ACM Transactions on Information Systems, vol. 9, no. 2, pages 99–122, April 1991.

[Chatty 04] Stephane Chatty, Stephane Sire, Jean-Luc Vinot, Patrick Lecoanet, Alexandre Lemort & Christophe Mertz. *Revisiting Visual Interface Programming : Creating GUI Tools for Designers and Programmers.* In Proceedings of the 17th annual ACM symposium on User interface software and technology, volume 6, pages 267–276, 2004.

[Cohen 89] Philip R. Cohen, M Dalrymple, D B Moran, F C Pereira & J W Sullivan. *Synergistic use of direct manipulation and natural language.* In Proceedings of the SIGCHI conference on Human factors in computing systems: Wings for the mind, CHI '89, pages 227–233, New York, NY, USA, 1989. ACM.

[Coutaz 87] Joëlle Coutaz. *PAC, an object oriented model for dialog design.* In Proceedings Interact, volume 87, pages 431–436, 1987.

[Coutaz 91] Joëlle Coutaz & Len Bass. Developing Software for the User Interface. Addison Wesley, 1991.

[Coutaz 93] Joëlle Coutaz, Laurence Nigay & Daniel Salber. *The AMODEUS Project ESPRIT Basic ResearchAction 7040.* Architecture, no. Cci, 1993.

[Coutaz 95] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May & Richard M. Young. *Four easy pieces for assessing the usability of multimodal interaction: the CARE properties.* Proceedings of INTERACT, vol. 95, no. June, pages 115–120, 1995.

[Coutaz 05] Joëlle Coutaz, James L Crowley, Simon Dobson & David Garlan. *Context key.* Communications of the ACM, vol. 48, no. 3, pages 49–53, 2005.

[Dragicevic 04] Pierre Dragicevic. *Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables*, 2004.

[Duke 93] D. Duke & M. Harrison. *Towards a Theory of Interactors.* The Amodeus Project, Esprit Basic Research, vol. 7040, 1993.

[Duke 95] DJ Duke & M.D. Harrison. *Event model of human-system interaction.* Software Engineering Journal, vol. 10, no. 1, pages 3–12, 1995.

[Efron 41] D. Efron. *Gesture and Environment. Morningside Heights.* NY: King's Crown Press. Republished 1972 as Gesture, Race, and Culture. The Hague: Mouton., 1941.

[Ekman 69] Paul Ekman & Wallace V. Friesen. *The Repertoire of Nonverbal Bahavior: Categories, Origins, Usage, and Coding.* Semiotica, 1969.

[Ekman 72] Paul Ekman & Wallace V. Friesen. *Hand Movements.* The journal of communication, pages 353–374, 1972.

[Faconti 93] GP Faconti. *Towards the concept of interactor.* Amodeus Project Document: System Modelling/WP8, pages 1–23, 1993.

[Fishkin 98] Kenneth P. Fishkin, Thomas P Moran & Beverly L Harrison. *Embodied User Interfaces: Towards Invisible User Interfaces.* In Proceedings of EHCI, pages 1–18, Deventer, The Netherlands, The Netherlands, 1998. Kluwer, B.V.

[Fitzmaurice 93] George W. Fitzmaurice, Shumin Zhai & Mark H. Chignell. *Virtual reality for palmtop computers.* ACM Transactions on Information Systems, vol. 11, no. 3, pages 197–218, July 1993.

[Foley 90a] James D. Foley, Andries van Dam, Steven K. Feiner & John F. Hughes. Computer graphics: principles and practice (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., 1990.

[Foley 90b] James D. Foley, Victor L. Wallace & Peggy Chan. *The human factors of computer graphics interaction techniques.* In Human-computer interaction, pages 67–121. Prentice Hall Press, 1990.

[Francone 09] Jérémie Francone, Gilles Bailly, Laurence Nigay & Eric Lecolinet. *Wavelet menu: une adaptation des marking menus pour les dispositifs mobiles.* In IHM '09: Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine, pages 367–370, New York, NY, USA, 2009. ACM.

[Grossman 04] Tovi Grossman, Daniel Wigdor & Ravin Balakrishnan. *Multi-finger gestural interaction with 3d volumetric displays.* In Proceedings of the 17th annual ACM symposium on User interface software and technology, UIST '04, pages 61–70, New York, NY, USA, 2004. ACM.

[Harrison 98] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon & Roy Want. *Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces.* Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '98, no. April, pages 17–24, 1998.

[Helander 97] Martin G. Helander, Thomas K. Landauer & Prasad V. Prabhu. Handbook of Human-Computer Interaction. Elsevier Science Inc., New York, NY, USA, 1997.

[Hemmert 08] Fabian Hemmert, Gesche Joost, André Knörig & Reto Wettach. *Dynamic knobs: shape change as a means of interaction on a mobile phone.* In CHI'08 extended abstracts on Human factors in computing systems, pages 2309–2314. ACM, 2008.

[Hinckley 00] Ken Hinckley, Jeff Pierce, Mike Sinclair & Eric Horvitz. *Sensing techniques for mobile interaction.* Proceedings of the 13th annual ACM symposium on User interface software and technology - UIST '00, vol. 2, pages 91–100, 2000.

[Hinckley 04] Ken Hinckley, R. Jacob & Colin Ware. *Input/output devices and interaction techniques*, 2004.

[Hinckley 11] Ken Hinckley & Hyunyoung Song. *Sensor Synaesthesia : Touch in Motion , and Motion in Touch.* In Proceedings of the 2011 annual conference on Human factors in computing systems, pages 801–810. ACM, 2011.

[Hutchins 85] Edwin L Hutchins, James D Hollan & Donald A. Norman. *Direct manipulation interfaces.* Hum.-Comput. Interact., vol. 1, no. 4, pages 311–338, December 1985.

[Ishii 97] Hiroshi Ishii & Brygg Ullmer. *Tangible bits: towards seamless interfaces between people, bits and atoms.* In Proceedings of the

SIGCHI conference on Human factors in computing systems, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

[Jacob 03] Robert J.K. Jacob & K.S. Karn. *Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises.* The Mind's eye: Cognitive The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research, pages 573–603, 2003.

[Jacob 08] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey & Jamie Zigelbaum. *Reality-Based Interaction : A Framework for Post-WIMP Interfaces.* In Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, 2008.

[Jakobsen 07] Mikkel Ronne Jakobsen & Kasper Hornaek. *Transient visualizations.* In OZCHI '07: Proceedings of the 19th Australasian conference on Computer-Human Interaction, pages 69–76, New York, NY, USA, 2007. ACM.

[Karam 05] Maria Karam & M. C. Schraefel. *A taxonomy of Gestures in Human Computer Interaction.* ACM Transactions on Computer-Human Interaction, pages 1–45, 2005.

[Kratz 09] Sven Kratz & Rafael Ballagas. *Unravelling seams: improving mobile gesture recognition with visual feedback techniques.* In Proceedings of CHI, volume 9, pages 937–940, 2009.

[Kurtenbach 93] Gordon Kurtenbach, Abigail Sellen & William Buxton. *An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus.* Human-Computer Interaction, vol. 8, no. 1, pages 1–23, March 1993.

[Lachenal 04] Christophe Lachenal. *Modèle et infrastructure logicielle pour l'interaction multi-instrument multisurface.* PhD thesis, Universitée Joseph Fourier, 2004.

[Levin 99] Golan Levin & Paul Yarin. *Bringing sketching tools to keychain computers with an acceleration-based interface.* CHI '99 extended abstracts on Human factors in computer systems - CHI '99, page 268, 1999.

[Liu 06] Yili Liu, Robert Feyen & Omer Tsimhoni. *Queueing Network-Model Human Processor (QN-MHP): A computational architecture for multitask performance in human-machine systems.* ACM Transactions on Computer-Human Interaction (TOCHI), vol. 13, no. 1, pages 37–70, 2006.

[Loehr 04] Daniel P. Loehr. *Gesture and Intonation.* PhD thesis, Georgetown University, 2004.

[Mackinlay 90] Jock Mackinlay, Stuart K. Card & George G. Robertson. *A semantic analysis of the design space of input devices.* Human–Computer Interaction, vol. 5, no. 2, pages 145–190, 1990.

[McNeill 92] David McNeill. Hand and Mind: What Gestures Reveal about Thought. University Of Chicago Press, 1992.

[Moran 81] Thomas P. Moran. *The Command Language Grammar: a representation for the user interface of interactive computer systems.* International Journal of Man-Machine Studies, vol. 15, no. 1, pages 3–50, July 1981.

[Morrel-Samuels 90] P Morrel-Samuels. *Clarifying the distinction between lexical and gestural commands.* Int. J. Man-Mach. Stud., vol. 32, no. 5, pages 581–590, May 1990.

[Nancel 09] Mathieu Nancel, Stéphane Huot & Michel Beaudouin-Lafon. *Un espace de conception fondé sur une analyse morphologique des techniques de menus.* In Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine, pages 13–22. ACM, 2009.

[Nigay 96] Laurence Nigay & Joëlle Coutaz. *Espaces conceptuels pour l'interaction multimédia et multimodale.* TSI, spécial Multimédia et Collecticiel, AFCET & Hermes Publ., vol. 15, no. 9, pages 1195–1225, 1996.

[Norman 86a] Donald A. Norman. User Centered System Design; New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc., 1986.

[Norman 86b] Donald A. Norman & Stephen W. Draper. User Centered System

Design; New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc., 1986.

[Norman 10] Donald A. Norman. *Natural User Interfaces Are Not Natural.* Human Interfaces, pages 6–10, 2010.

[Olwal 03] Alex Olwal & Steven Feiner. *Rubbing the Fisheye: Precise Touch-Screen Interaction with Gestures and Fisheye Views.* In ext. abst. UIST'03, pages 83–84, 2003.

[Olwal 08] Alex Olwal, Steven Feiner & Susanna Heyman. *Rubbing and tapping for precise and rapid selection on touch-screen displays.* Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08, vol. 2008, page 295, 2008.

[Oviatt 92] Sharon Oviatt. *Pen/voice: Complementary multimodal communication.* Proceedings of SpeechTech, vol. 92, pages 238–241, 1992.

[Parhi 06] Pekka Parhi, Amy K. Karlson & Benjamin B. Bederson. *Target size study for one-handed thumb use on small touchscreen devices.* Proceedings of the 8th conference on Human-computer interaction with mobile devices and services - MobileHCI '06, page 203, 2006.

[Partridge 02] Kurt Partridge, Saurav Chatterjee, Vibha Sazawal, Gaetano Borriello & Roy Want. *TiltType: accelerometer-supported text entry for very small devices.* Proceedings of the 15th annual ACM symposium on User interface software and technology, vol. 4, no. 2, pages 201–204, 2002.

[Patel 04] Shwetak N. Patel, Jeffrey S. Pierce & Gregory D. Abowd. *A gesture-based authentication scheme for untrusted public terminals.* Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04, vol. 6, no. 2, page 157, 2004.

[Potter 88] Richard L. Potter, Linda J. Weldon & Ben Shneiderman. *Improving the accuracy of touch screens: an experimental evaluation of three strategies.* Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88, pages 27–32, 1988.

[Quek 02] Francis Quek, David McNeill, Robert Bryll, Susan Duncan, Xin-Feng Ma, Cemil Kirbas, Karl E. McCullough & Rashid Ansari.

Multimodal human discourse: gesture and speech. ACM Transactions on Computer-Human Interaction (TOCHI), vol. 9, no. 3, pages 171–193, 2002.

[Raskin 00] Jef Raskin. The humane interface: new directions for designing interactive systems. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[Rekimoto 96] Jun Rekimoto. *Tilting operations for small screen interfaces.* Proceedings of the 9th annual ACM symposium on User interface software and technology - UIST '96, pages 167–168, 1996.

[Rico 10] Julie Rico & Stephen Brewster. *Usable gestures for mobile interfaces: evaluating social acceptability.* In Proc. CHI, volume 10, 2010.

[Roudaut 08] Anne Roudaut, Stéphane Huot & Eric Lecolinet. *TapTap and MagStick: improving one-handed target acquisition on small touch-screens.* In Proceedings of the working conference on Advanced visual interfaces, pages 146–153. ACM, 2008.

[Roudaut 09] Anne Roudaut, Mathias Baglioni & Eric Lecolinet. *TimeTilt: Using Sensor-Based Gestures to Travel through Multiple Applications on a Mobile Device.* In INTERACT '09: Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction, pages 830–834, Berlin, Heidelberg, 2009. Springer-Verlag.

[Roudaut 10] Anne Roudaut. *Conception et Evaluation de Techniques d ' Interaction pour Dispositifs Mobiles.* PhD thesis, Telecom ParisTech, 2010.

[Roudaut 11] Anne Roudaut, Henning Pohl & Patrick Baudisch. *Touch input on curved surfaces.* In Proceedings of the 2011 annual conference on Human factors in computing systems, CHI '11, pages 1011–1020, New York, NY, USA, 2011. ACM.

[Rubine 91] Dean Rubine. *Specifying gestures by example.* ACM SIGGRAPH Computer Graphics, vol. 25, no. 4, pages 329–337, July 1991.

[Schwesig 04] Carsten Schwesig, Ivan Poupyrev & Eijiro Mori. *Gummi: a bendable computer.* Proceedings of the SIGCHI conference on

Human factors in computing systems, vol. 6, no. 1, pages 263–270, 2004.

[Sears 91] Andrew Sears & Ben Shneiderman. *High precision touchscreens: design strategies and comparisons with a mouse*. Int. J. Man-Mach. Stud., vol. 34, no. 4, pages 593–613, April 1991.

[Sellen 90] Abigail Sellen, Gordon Kurtenbach & William Buxton. *The role of visual and kinesthetic feedback in the prevention of mode errors*. In Proc. INTERACT'90, pages 667–673. IFIP, 1990.

[Shaw 03] Mary Shaw. *Writing good software engineering research papers*. 25th International Conference on Software Engineering, 2003. Proceedings., pages 726–736, 2003.

[Shneiderman 87] Ben Shneiderman. Designing the User Interface, chapitre 3 Menu Sel, pages 85–133. Addison-Wesley Publishing Company, 1987.

[Shoemaker 07] Garth Shoemaker, Anthony Tang & Kellogg S Booth. *Shadow reaching: a new perspective on interaction for large displays*. In Proceedings of the 20th annual ACM symposium on User interface software and technology, UIST '07, pages 53–56, New York, NY, USA, 2007. ACM.

[Swick 88] Ralph R Swick & Mark S Ackerman. *The X Toolkit: More Bricks for Building User Interfaces, or Widgets For Hire*. In USENIX Technical Conference, 1988.

[Taylor 08] Brandon Taylor & V. Michael Bove Jr. *The bar of soap: a grasp recognition system implemented in a multi-functional handheld device*. In CHI'08 extended abstracts on Human factors in computing systems, pages 3459–3464. ACM, 2008.

[Villamore 10] Craig Villamore, Dan Willis & Luke Wroblewski. *Touch Gesture reference guide*, 2010.

[Vogel 07] Daniel Vogel & Patrick Baudisch. *Shift: a technique for operating pen-based interfaces using touch*. In Proceedings of the SIGCHI conference on Human factors in computing systems, numéro c, pages 657–666. ACM, 2007.

[Wigdor 07] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell & Chia Shen. *LucidTouch: A See-Through Mobile Device*. UIST 2007, pages 269–278, 2007.

[Williamson 07] John Williamson, Rod Murray-Smith & Stephen Hughes. *Shoogle: excitatory multimodal interaction on mobile devices*. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 121–124. ACM, 2007.

[Wilson 03] Andrew D. Wilson & Steven Shafer. *XWand: UI for intelligent spaces*. In Proceedings of the SIGCHI conference on Human factors in computing systems, numéro 5, page 552. ACM, 2003.

[Wobbrock 07] Jacob O. Wobbrock, Andrew D. Wilson & Yang Li. *Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes*. In Proceedings of the 20th annual ACM symposium on User interface software and technology, pages 159–168. ACM, 2007.

[Wobbrock 09] Jacob O. Wobbrock, Meredith Ringel Morris & Andrew D. Wilson. *User-defined gestures for surface computing*. In Proceedings of the 27th international conference on Human factors in computing systems, pages 1083–1092. ACM, 2009.

[Yatani 08] Koji Yatani, Kurt Partridge, Marshall Bern & Mark W. Newman. *Escape: a target selection technique using visually-cued gestures*. In Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, numéro c, pages 285–294. ACM, 2008.

[Yee 03] K.P. Yee. *Peephole displays: pen interaction on spatially aware handheld computers*. In Proceedings of the SIGCHI conference on Human factors in computing systems, numéro 5, pages 1–8. ACM, 2003.

Abstract

In this thesis, I address the question of gestural interaction on mobile devices. These, now common, differ from conventional computers primarily by the input devices the user interact with (small screen size but tactile, various sensors such as accelerometers) as well as the context in which they are used. The work I present is an exploration of the vast area of interaction techniques on these mobile devices. I structure this space by focusing on the techniques based on accelerometers for which I propose a taxonomy. Its descriptive and discriminant power is validated by the classification of thirty-seven interaction techniques in the literature. The rest of my work focuses on the achievement of gestural interaction techniques for these mobile devices. With TouchOver, I show that it is possible to take advantage of complementary two-channel input (touch screen and accelerometers) to add a state to the finger-drag, thus enriching the interaction. Finally, I focus on mobile device menus and I propose a new form of sign language menus. I discuss their implementation with the GeLATI software library that allows their integration into a pre-existing GUI toolkit.

Keywords
Human Computer Interaction, Gestural Interaction Techniques, Gestural Menus.

Résumé

Dans cette thèse, j'aborde la question de l'interaction gestuelle sur dispositif mobile. Ces dispositifs, à présent communs, se distinguent des ordinateurs conventionnels principalement par leurs périphériques d'interaction avec l'utilisateur (écrans de taille restreinte mais tactiles, capteurs divers tels que les accéléromètres) ainsi que par le contexte dans lequel ils sont utilisés. Le travail que je présente est une exploration du vaste domaine des techniques d'interaction sur ces dispositifs mobiles. Je structure cet espace en me concentrant sur les techniques à base d'accéléromètres pour lesquelles je propose une taxonomie. Son pouvoir descriptif et discriminant est validé par la classification de trente-sept techniques d'interaction de la littérature. La suite de mon travail se penche sur la réalisation de techniques d'interaction gestuelles pour ces dispositifs mobiles. Avec TouchOver, je montre qu'il est possible de tirer parti de manière complémentaire de deux canaux d'entrée (écran tactile et accéléromètres) pour ajouter un état au glissé du doigt, permettant ainsi d'enrichir cette interaction. Enfin, je m'intéresse aux menus sur dispositif mobile et je propose une nouvelle forme de menus gestuels. Je présente leur réalisation avec la bibliothèque logicielle GeLATI qui permet leur intégration à une boîte à outils de développement d'interface graphique préexistante.

Mots-clés
Interaction Homme-Machine, Techniques d'Interaction Gestuelle, Menus Gestuels.