DISS. ETH NO. 21686

Towards Document Engineering on Pen and Touch-Operated Interactive Tabletops

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

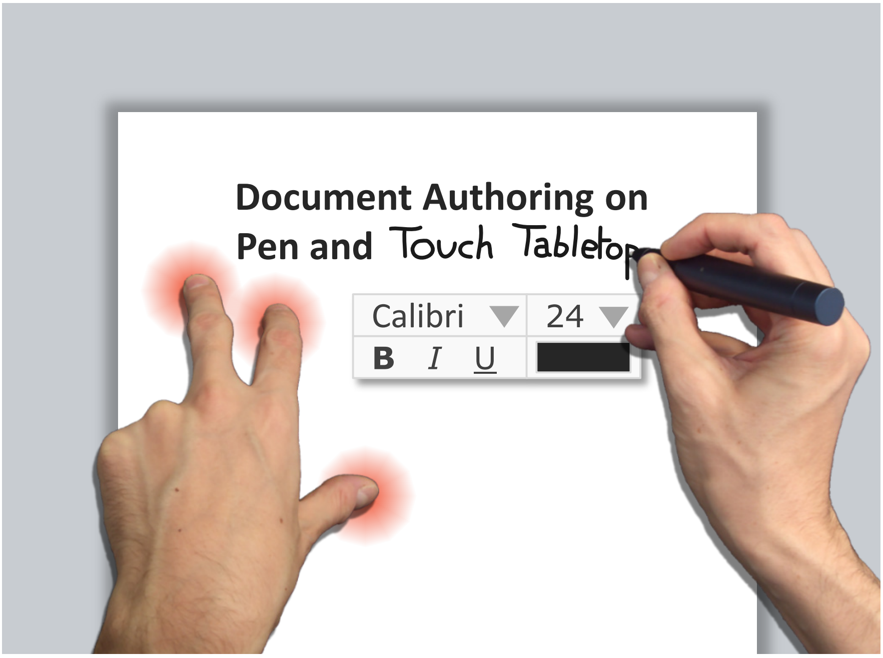
FABRICE MATULIC

Diplôme d'Ingénieur ENSIMAG
Diplom Informatik, Universität Karlsruhe (TH)

accepted on the recommendation of

Prof. Dr. Moira C. Norrie, examiner
Prof. (FH) Dr. Michael Haller, co-examiner
Dr. Ken Hinckley, co-examiner

2014


Document Authoring on
Pen and Touch Tabletop
Calibri
24
B I U

Abstract

In the last few years, the consumer device landscape has been significantly transformed by the rapid proliferation of smartphones and tablets, whose sales are fast overtaking that of traditional PCs. The highly user-friendly multitouch interfaces combined with ever more efficient electronics have made it possible to perform an increasingly wide range of activities on those mobile devices. One category of tasks, for which the desktop PC with mouse and keyboard remains the platform of choice, however, is office work. In particular, productivity document engineering tasks are still more likely to be carried out on a regular computer and with software, whose interfaces are based on the time-honoured "windows, icons, menus, pointer" (WIMP) paradigm. With the recent emergence of larger touch-based devices such as wall screens and digital tabletops (which overcome the limitations of mobile devices in terms of available screen real estate) as well as the resurgence of the stylus as an additional input instrument, the vocabulary of interactions at the disposal of interface designers has vastly broadened. Hybrid pen and touch input is a very new and promising interaction model that practitioners have begun to tap, but it has yet to be applied in office applications and document-centric productivity environments.

Within the ongoing quest to conceive the office of the future, the goal of this work is to show how the interactive potential of pen and touch-operated tabletops as digital work desks can be leveraged to execute essential document engineering tasks such as document editing and authoring.

In a first step, important properties of pen and touch input such as docking and tracing precision, palm rejection, mode preference for selections and bimanual coordination, are analysed and experimental results are provided, adding to the –as of yet, relatively scant– body of empirical data on the subject. One powerful asset of the pen and touch interaction paradigm that especially stands out in the experiments is the possibility to rapidly and smoothly switch the pen's function using the non-dominant hand (NDH). This creates an interesting design space, in which particular quasimodal (i.e. maintained) NDH postures on the touch-sensitive surface such as chords and recognisable contact shape patterns can be associated with rich categories of gestural commands articulated by the pen.

In order to support developers in the creation of pen and touch applications with tight integration and interplay of the two modalities, and after realising that current major APIs are largely inadequate, a dedicated software framework is designed and implemented. At the lowest level, the framework provides an abstraction of hardware-specific input data, especially taking into account the limitations of the multitouch tabletop system used as enabling hardware: the DiamondTouch. In the middle layer, it integrates a powerful gesture module defining a set of handy pen and touch gestures, with flexible constructs to implement custom extensions. At the top level, the framework includes a component hierarchy with built-in UI elements endowed with typical behaviours controlled by classic pen and touch interactions.

Based on this framework, two application prototypes are implemented with two carefully chosen document scenarios in mind, considering, among other factors, the suitability of particular types of documents for pen and touch interaction: active reading, and authoring of presentational documents. In the first program, an approach following a desk/codex metaphor, but augmented with tools from the digital world, is tested. Hence, while documents are shown as bound volumes, whose pages can be "flipped", users can also perform text searches with handwritten keywords and make use of various pen and touch functions for non-sequential document navigation. The user evaluation provides some insights as to the suitability of the tabletop platform for active reading tasks and how far it is able to combine the advantages of the analogue and digital worlds without the disadvantages of each.

The second application, which allows users to create rich documents from scratch or based on existing documents, is the main contribution of the thesis. Its design relaxes the desk/paper metaphor in favour of a more pragmatic approach and a more extensive reliance on gestures (uni- and bimanual) instead of widgets. Those gestures are divided into different inking and non-inking modes activated by specific NDH postures. Modes in the former category consist of text and shape entry, where the system recognises the user's freeform input and converts it to formatted text or shape data. The non-inking modes associate groups of pen and touch gestures with common document-editing operations to add, select, edit and style content. Finally, the application integrates a clipart-search tool, which, in addition to regular keyword-searching also supports sketch-based queries. A lab study confirms the potential of this novel document-authoring environment and identifies areas of improvement for future iterations of the system.

The thesis concludes with an overarching discussion of important issues that still need to be overcome in order to further the case of digital tabletops as compelling platforms for document engineering tasks and offers a vision of how the interactive work desk might make its way into the offices of the (near) future, among the other digital revolutions that lie ahead.

Résumé

Ces dernières années ont vu la transformation radicale du parc des appareils électroniques grand-public, sous l'impulsion des smartphones et des tablettes tactiles dont les ventes sont en passe de dépasser celles des PCs traditionnels. Les interfaces "multitouch" plus accessibles et plus faciles d'utilisation, couplées aux progrès de l'électronique ont permis à un nombre d'activités toujours plus grand d'être accomplies sur ces appareils mobiles. Une certaine catégorie de tâches pour laquelle le PC équipé de souris et de clavier demeure néanmoins la plateforme de choix est la bureautique. En particulier, les tâches liées à la production de documents sont plus à même d'être effectuées sur un ordinateur de bureau avec des logiciels dont les interfaces graphiques sont basées sur le principe consacré WIMP, c'est-à-dire avec des fenêtres, des icônes, des menus et un pointeur. Avec l'émergence des appareils tactiles disposant d'une plus large surface d'interaction, tels que les écrans muraux et les tables digitales, et la réapparition du stylet comme instrument de saisie complémentaire, le vocabulaire d'interactions à la disposition des concepteurs d'interfaces graphiques s'est considérablement agrandi. Le nouveau modèle d'interaction combinant le toucher et le stylo digital que les chercheurs et ingénieurs ont commencé à exploiter est prometteur, mais il n'a pas encore été mis en pratique pour les applications bureautiques et le traitement de documents.

Dans le cadre de la quête perpétuelle qui vise à concevoir le bureau du futur, l'objectif de ce travail est de montrer comment le potentiel des tables interactives, contrôlées par le toucher et le stylet, peut être exploité pour le traitement et la production de documents, en particulier leur édition et création.

Dans un premier temps, des propriétés fondamentales de ce modèle d'interaction, telles que la précision de traçage et de positionnement, les interférences de la paume de la main qui écrit avec la surface tactile, les préférences de modes pour les opérations de sélection et la coordination bimanuelle sont analysées et des résultats expérimentaux sont présentés. Ces-derniers viennent étoffer le volume, à ce jour bien maigre, de données empiriques disponibles dans la littérature scientifique concernant ce moyen d'interaction. L'un des atouts majeurs de la saisie au stylo et au toucher qui ressort des expériences est la possibilité de modifier rapidement et facilement la fonction du stylet via la main non-dominante (MND). Cette faculté permet notamment d'associer des poses quasimodales (c'est-à-dire maintenues sur la surface) de la MND, telles que des combinaisons de doigts ou des positions de mains particulières (postures), à de riches catégories de commandes gestuelles effectuées par le stylo (modes).

Afin de faciliter la tâche des développeurs d'application qui s'appuient sur la gestion simultanée du toucher et du stylet et après avoir constaté l'inadéquation des APIs actuelles, une infrastructure logicielle dédiée est conçue et réalisée. Dans sa couche bas-niveau, la bibliothèque définit un dénominateur commun aux multiples types de signaux d'entrées spécifiques aux différents appareils et propose une interface universelle pour la gestion de ces données par les couches logicielles supérieures. Un soin particulier est pris pour prendre en

compte les spécificités de la table multitouch employée pour le développement de prototypes: la DiamondTouch. La couche logicielle intermédiaire intègre un module de gestion de gestes combinant les deux modalités avec un certain nombre de gestes standards prédéfinis et la possibilité pour les développeurs d'étendre leurs fonctionnalités jusqu'à concevoir leurs propres gestes bimodaux. Au niveau supérieur, le logiciel inclut une hiérarchie de composants graphiques qui incorporent des logiques de fonctionnement de base répondant aux interactions tactiles et du stylet.

Sur la base de cette fondation logicielle, sont développés deux prototypes répondant à deux scénarios de traitement de documents bien choisis (notamment en optant pour les types de documents qui semblent les mieux adaptés à ce genre d'interface): la lecture active et la création de documents de présentation. Dans le premier programme, l'interface graphique se présente comme un bureau virtuel avec des documents en pseudo 3D que les utilisateurs manipulent presque comme des documents réels en "tournant" les pages avec les doigts et en écrivant dessus avec le stylo. L'interface est complémentée par des outils typiques du monde numérique tels que la recherche texte et des fonctions permettant l'accès rapide aux informations contenues dans les documents. L'évaluation du prototype donne des indications précieuses sur l'aptitude à l'usage des tables interactives pour ce genre de tâche et leurs capacités à réunir les avantages des milieux analogique et numérique sans leurs désavantages.

La seconde application qui permet de créer des documents à partir de zéro ou à partir d'autres documents est la contribution principale de cette thèse. Le modèle de conception de son interface reprend certains éléments du prototype précédent mais suit une approche plus pragmatique avec un plus grand recours aux gestes (uni- et bimanuels) au lieu d'outils graphiques explicites. Ces gestes s'exécutent en activant différents modes, via des postures particulières de la MND. Le programme inclut deux modes de saisie directe, où le stylo marque directement les pages pour insérer texte et formes vectorielles par voie manuscrite (et convertis automatiquement par le système), et trois modes "augmentés" qui permettent de réaliser diverses opérations de sélection et d'édition via des commandes gestuelles. Enfin, l'application intègre un outil de recherche de cliparts par croquis en plus de la recherche classique par mots clés. Une étude en laboratoire confirme le potentiel de ce type de système pour la conception de documents et met en lumière certaines pistes d'améliorations pour les prochaines itérations du système.

La thèse conclut sur une discussion des problématiques principales rencontrées lors du projet, et expose le chemin encore à parcourir pour que les tables interactives puissent être adoptées comme instruments bureautiques intuitifs et performants dans les bureaux du futur (proche), parmi les autres révolutions du monde numérique que le progrès engendrera.

Acknowledgments

First of all, I would like to express my gratitude to Prof. Moira C. Norrie for giving me the opportunity to work in her group and for her trust and open mind, which allowed me to carry out tabletop-related research freely and autonomously. Thanks to her as well, I was able to attend numerous international conferences and visit other research laboratories, which, in addition to contributing to shape this project, allowed me to make and nurture many contacts within the HCI research community.

Then, I wish to warmly thank my two co-examiners Prof. Michael Haller and Dr. Ken Hinckley, who accepted to peruse and review this dissertation. Additionally, I am grateful to Michael for providing us the Anoto foil and the streaming Anoto pens, as well as for his valuable comments and help when editing the AVI paper. As for Ken, I thank him for giving me the opportunity to work with him at Microsoft Research, in Redmond, during the summer of 2013.

I would also like to thank my colleagues, students and the members of ETH staff, who contributed one way or another to the successful completion of this research effort, whether by participating in my user studies, by providing administrative or technical support or in many cases both. I am particularly indebted to Beate Bernhard, our indefatigable secretary, who always showed patience and diligence when I plied her with all sorts of questions and requests on administrative matters (to say nothing of the work that ensued from them). On the technical side, I am obliged to Matthias Geel and Maximilian Speicher for their software contributions, which facilitated the integration of the Anoto pen in the framework.

Much of the inspiration for this project and initial ideas about a tabletop-based document editor emerged from previous work I did at Ricoh and so I wish to express my recognition to my former employer and colleagues for immersing me in the fascinating world of document engineering and stylus-operated document systems. It is largely thanks to the enthusiastic response I received to SmartPublisher that I was encouraged to further explore that area.

Last but not least, I thank my family and friends, who, in addition to their relentless support, have always patiently listened and made an effort to understand when I tried to explain to them what on earth it is I was working on during those years.

Table of Contents

1 Introduction 1

1.1 Digital Documents and Their Environments 2

1.2 The Enhanced Workdesk in Research 5

1.3 Motivation 7

1.4 Contributions 11

1.5 Outline 13

2 Background 16

2.1 The Vision of the Office of the Future 16

2.2 Interactive Tabletops and Workdesks 19

2.2.1 Historical Interactive Tabletops 20

2.2.2 Non-Planar Desks 24

2.2.3 Modern Augmented Workdesks 25

2.3 Pen-Based Document Interaction 26

2.3.1 Pen-Based Computing 26

2.4 Pen and Touch Interaction 33

2.4.1 Asymmetric Bimanual Interaction 34

2.4.2 The Pen and Touch Interaction Paradigm 35

2.4.3 Pen and Touch-Enabled Apparatuses 37

2.4.4 Pen and Touch Applications 39

2.5 Analysis 42

3 Pen & Touch Software Framework 45

3.1 Requirements 46

3.1.1 Goals and Scope 46

3.1.2 Formulation of the Requirements 47
3.2 Architecture 48
3.2.1 Raw Input 48
3.2.2 Input Abstraction 50
3.2.3 Input Dispatching 52
3.2.4 Managing Layer 53
3.2.5 The Component Layer 59
3.3 Implementation 64
3.3.1 Hardware and I/O 64
3.3.2 Software Platform 65
3.4 Summary 65
4 Empirical Evaluation of Pen & Touch Interaction Properties 68
4.1 Related Studies 68
4.2 Study design and environment 69
4.2.1 Study Design 70
4.2.2 Participants 70
4.3 Experiment 1 71
4.3.1 Description 71
4.3.2 Results 74
4.4 Experiment 2 77
4.4.1 Description 77
4.4.2 Results 79
4.5 Experiment 3 82
4.5.1 Description 82
4.5.2 Results 85
4.6 Experiment 4 87
4.6.1 Description 87

4.6.2 Results 89

4.7 Conclusion 92

5 Document Scenario 1: Active Reading 95

5.1 Related Work on Active Reading 96

5.2 The System 97

5.2.1 Design Approach 97

5.2.2 User Interface 98

5.2.3 Gesture Set 100

5.2.4 Software Implementation 105

5.3 User Evaluation 106

5.3.1 Study Design 106

5.3.2 Description of the Tasks 107

5.3.3 Results 108

5.4 Enhancements 113

6 Document Scenario 2: Document Authoring 117

6.1 Design Approach 118

6.1.1 Gesture Model 118

6.1.2 Scope Definition 120

6.2 Application 122

6.2.1 Workspace 122

6.2.2 Global Modes 124

6.2.3 Stroke Input and Styling 125

6.2.4 Shape and Freeform Input 129

6.2.5 General Commands 130

6.2.6 Content Registration and Insertion 131

6.2.7 Clipart Retrieval with Query-by-Sketching 132

6.2.8 Function/Gesture Summary 133

6.3 Software Implementation 135

6.3.1 Document Model 136

6.3.2 UI and Interactions 136

6.4 Lab Study 137

6.4.1 Goals 137

6.4.2 Results 138

6.4.3 Discussion 141

6.5 Conclusion 142

7 Conclusions 144

7.1 Summary 144

7.1.1 Review 144

7.1.2 Summary of the Main Contributions 148

7.2 Discussion 149

7.3 Vision 151

1 Introduction

Documents, in a broad sense, are instruments or media that convey information for a specific purpose in a more or less lasting form. Traditionally, a document has been understood to mean any record carrying written or graphic information such as inked paper, but this definition has evolved in the last century to include more general and theoretical constructs. With the development of information science and "documentalisation", documents ceased to be defined solely as textual records and became instead evidence of something that has been observed or expressed. Paul Otlet, one of the pioneers of information science, noted that a variety of three-dimensional objects such as sculptures, artefacts and other works of art could also be considered documents, since they are an "expression of human thought" [149]. Following on this extrapolation, Suzanne Briet, or "Madame Documentation" as she was known, famously remarked that an antelope also qualified as a document once it had become an object of study or physical evidence of specific events (e.g. a capture and placement in a zoo) [46]. This question of what actually constitutes a document is in fact still an open question, which is why it has caught the interest of information scientists and historians. Michael Buckland in his seminal papers on the topic showed that the answer is indeed not obvious [48, 50], which has led Levy to offer the more open view that a document is simply "a way to delegate the ability to speak to inanimate objects" [122].

With the advent of the modern computer age, the concept of a document has further expanded to include other forms and media. In particular, documents can now exist in non-physical form and be unbounded, which reinforces the functional approach that needs to be embraced in order to capture the essence of a document [49]. A digital document can therefore be a digitised version of a physical document, but also, and perhaps more significantly, any kind of structured content created on a computer (it is interesting to notice that even dynamic containers of information such as web pages are also referred to as documents). This increased complexity and diversity of the document landscape is matched by the sophistication of the processes that govern the production, manipulation and maintenance of these documents. Beyond the societal concerns raised for instance by Levy [122], there are many technical challenges that came to light with the technological revolution, from providing suitable models of documents to designing systems that process and output said documents in a timely and efficient manner.

The study and solving of the aforementioned problems are one of the main goals of the discipline of "document engineering", which emerged a little over a decade ago [1]. In their book "Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services", Glushko and McGrath describe this special research field as a synthesis of "complementary ideas from information and systems analysis, electronic publishing, business process analysis, and business informatics." [79] To this list, one can add the study of human factors that play a vital role in document processes, considering many –if not most– of them require interaction with people at some point. In particular, the creation process, i.e. the authoring and editing of documents, needs to be supported by adequate tools so that users can efficiently engage with documents without being burdened by the rigidity and frustrations of badly designed systems. While it is relatively simple to develop a bare-bones editor for plain non-formatted text, creating a comprehensive authoring platform for the composition of complex documents with rich layouts and styles is a different kettle of fish. This task is compounded by modern-day requirements which demand that those tools be both efficient and accessible.

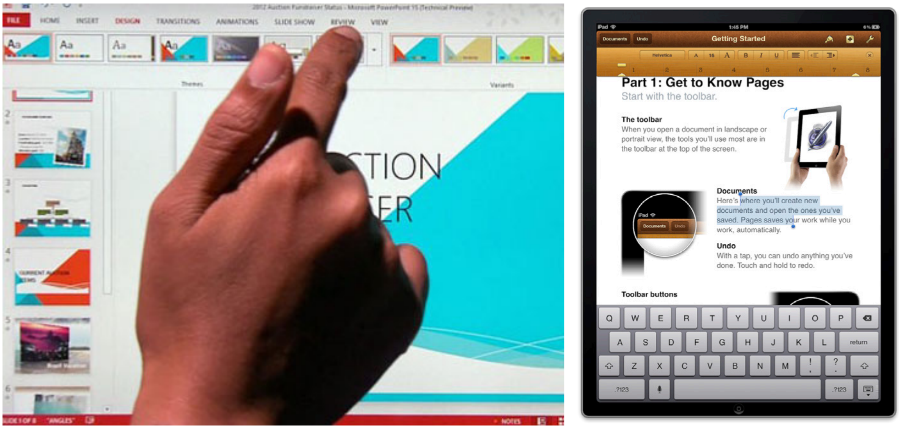
1.1 Digital Documents and Their Environments

The proliferation of authoring devices and the increasing democratisation of publishing tools has ensured that production of content has become easier and cheaper than ever. While it is difficult to estimate the number of digital documents that are created or edited daily, one can safely say that document-related tasks are extremely common in the present modern world, both within and outside the circle of professional activities. From reports to presentations, novels to works of art, news articles to this thesis etc. an unfathomable number of documents are authored today, most of which, using computers and electronic devices. There is indeed a steady demand for word processors, desktop publishing software (DTP[1]) and other content-creation utilities. In mid-2012, for instance, Microsoft's Office division asserted that its popular suite, which includes the widely used word processor Word and presentation software PowerPoint among other programs, was "used by more than a billion people" [16]. This claim might seem somewhat optimistic, but it gives an idea of the extent to which digital office documents are even more relevant nowadays. With respect to presentations in particular, the company estimated more than a decade ago that 30 million PowerPoint presentations were created every day [151], a number that likely has increased since.

This explosion of document production has been facilitated by increasingly sophisticated authoring tools, whose graphical user interfaces, for the past 30 years, have relied upon the

[1] http://en.wikipedia.org/wiki/Desktop_publishing

popular WIMP [2] paradigm. Since the early days of WYSIWYG [3] word processors, the computer+keyboard+mouse trio has been the dominating instrument to perform document-related work, but the rapidly expanding popularity of NUI[4]-based devices of the last few years has begun to challenge the status quo. Driven by smartphones, tablets and surface computers, multitouch has now become a mainstream interaction method that is being considered for a wide variety of tasks in different environments, including document engineering activities. The latest iteration of Microsoft Office follows the new path taken by Microsoft Windows 8 by adopting touch-friendly components, although the interface is still far from having shed its WIMP origins [47]. Microsoft's main competitor Apple has also ported its own desktop office suite iWork [5] to the iOS platform at the core of its mobile products (Figure 1.1).



Part 1: Get to Know Pages
Start with the toolbar.

The toolbar
When you open a document in landscape or portrait view, the tools you'll use most are in the toolbar at the top of the screen.

Documents
Here's where you'll create new documents and open the ones you've saved. Pages saves your work while you work, automatically.

Undo
With a tap, you can undo anything you've done. Touch and hold to redo.

Toolbar buttons

Figure 1.1: Mobile office suites on tactile screen devices: Microsoft Powerpoint, Office 2013 (left) and Apple Pages, iWork (right)

Further underlining the trend towards more natural interfaces is the resurgence of stylus-operated devices such as tablets and e-book readers, which promise an experience close to that of pen and paper. Machines produced nowadays have gone a long way since the early days of pen-computing more than 40 years ago (interestingly, pen computing has a long history, which predates the mouse, as the first tablet-controlled system with handwriting recognition was demonstrated to the public in 1957 [66]). Current tablets boast increased sensing resolutions and are able to react to speed, pressure and hovering. Yet despite the progress made by pen digitisers and the apparent benefits that these systems offer, they have mostly remained a niche market, one can surmise, because hardware is still unable to completely match the affordances of paper and also because most work can be efficiently executed with other instruments.

[2] http://en.wikipedia.org/wiki/WIMP_%28computing%29
[3] http://en.wikipedia.org/wiki/WYSIWYG
[4] http://en.wikipedia.org/wiki/Natural_User_Interface

Notwithstanding, the stylus has consistently remained a popular choice for particular tasks such as art creation, CAD and note-taking.

Along with tablet computers, there has also been a parallel effort of augmenting the capabilities of analogue paper in order to endow it with some of the conveniences of the digital world (e.g. persistent storage, content linking). The company Anoto, for instance, developed a technology that allows a digital pen with an embedded camera to track its position on a regular piece of paper, thanks to a unique dot pattern printed on it [3]. This "interactive paper" has spawned a number of applications in business and educational settings to capture content such as notes and sketches that can be synchronised with a computer system [4]. In the research community, the technology has been applied in comprehensive frameworks for the development of interactive paper applications [147, 211]. Notable examples of the latter include a paper-based presentation tool [176], a gesture-based system to link paper documents with digital content [127] and a proof-editing application to edit digital documents on paper [196]. Plain paper approaches, however, have the drawback that they lack immediate feedback, which is often essential in highly interactive contexts. Solutions resorting to projectors [84, 177] and pen-mounted lights [128] have been attempted, but they usually require more sophisticated setups than a simple stylus-operated tablet.

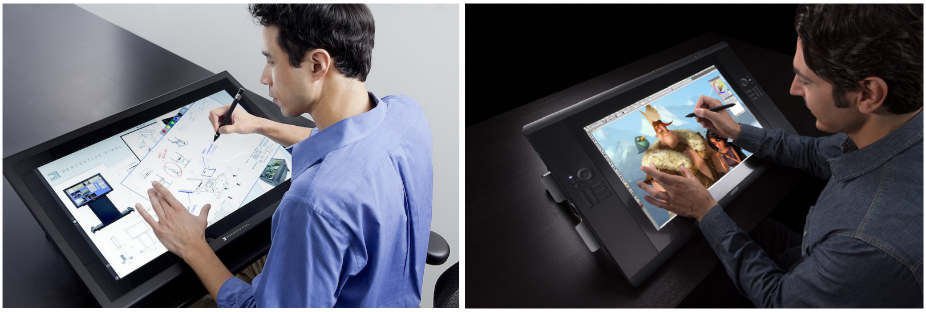


Figure 1.2: Two commercial surface systems with support for simultaneous pen and touch: Perceptive Pixel's Active Stylus screen (left) and Wacom's Cintiq 24HD Touch (right)

Until recently, manufacturers of keyboardless devices relied on either pen or touch as input method, but not both. This was partly due to sensing technology not being able to differentiate between the two input types, but also because for a long time, pen and touch were considered alternative and not complementary interaction methods on consumer devices. Pen support was eventually added to some touch screens, but this came with limitations, as either the pen was captured as an additional finger (albeit a much thinner one) or the two modalities were not available at the same time. The strict separation of pen and touch input is in fact standard on current Windows and Android systems (typically, touch is entirely deactivated when the stylus comes within range of the screen). It is only in 2012 that products with built-in support for simultaneous and differentiated pen and touch input came onto the market [17, 19, 23] (Figure 1.2). Those systems are still very new and relatively expensive, however. This means that there are virtually no applications that fully take advantage of the increased interactive bandwidth afforded by those devices on the market yet.

1.2 The Enhanced Workdesk in Research

Well before the release of commercial solutions, HCI researchers and practitioners have grappled with the problem of creating user interfaces to interact with digital and analogue content in a more natural way. Since Wellner's pioneering Digital Desk developed at Xerox EuroPARC in the early 90's [198], many experimental prototypes seeking to enhance the workdesk have been produced. A number of experimental apparatuses followed Wellner's approach of augmenting a regular office desk with vision-based systems such as cameras and projectors to detect the location of objects or the user's fingers [30, 67, 109, 114, 162, 202, 206]. The main disadvantages of such systems are that they are prone to tracking errors due to occlusions and misdetections. They also typically suffer from latency problems, due to the expensive computer vision processes involved in capturing and analysing the large amount of raw input data. Moreover, they usually cannot differentiate between users. Those considerations along with technological improvements motivated researchers to venture into the realm of virtual reality. Specifically, they investigated how far the physical work environment could be modelled in a virtual interface so that people could directly work on a computer but still take advantage of their spatial cognition abilities to manage and organise information in 2D or 3D space. A few historical projects belonging to this category include Workscape [34], Data Mountain [164], Task Gallery [165] and BumpTop [26]. Studies have shown, however, that those types of interfaces have limited benefits and that increasing people's freedom to spatially arrange information in full 3D has, at best, no influence [57] and at worst, a negative impact [60] on retrieval efficiency.

With the appearance of large screens with built-in touch sensing, researchers have increasingly turned to the latter as enabling hardware for their experiments. Since the Digital Desk, numerous experimental systems have been built, some of which were later turned into commercial products. In chronological order, one can cite for example the Active Desk (1992) [52], the metaDESK (1997) [98, 192], the DiamondTouch (2001) [65], SmartSkin (2002) [161], reacTable (2003) [101], Han's FTIR tables (2005) [86] and PlayAnywhere (2005) [202]. The most widely known tabletop computer today, however, is no doubt Microsoft's PixelSense (formerly called "Surface") [15]. Currently in its second version, the table manufactured by Samsung is built around an LCD panel enhanced with IR sensors to detect objects placed on its surface.

There is a sizeable body of work that involves interactive tables, from the creation of experimental systems to the development of applications and studies investigating various aspects of their use. A comprehensive list with a taxonomy of major tabletop systems can be found in [115].

With respect to simultaneous pen and touch support, one of the earliest prototypes with integrated digitisers appeared in 2004 [209] (a version of the Active Desk did have some level of bimanual touch/stylus interaction, but it was not fully capable of using finger touches as the system was only able to detect specific hand postures via an overhead camera [52]). It combined a stylus-operated tablet PC and a resistive touch panel. This approach of supplementing an existing unimodal surface with a different digitising technology supporting the other input type was subsequently adopted by most researchers for their pen and touch systems. An easy solution, chosen by many, consists in layering a sheet with an Anoto pattern over a touchscreen such as a DiamondTouch [65]. This allows a clear separation between the two input methods at the software level and hence facilitates application development. Other techniques using a single digitiser rely on segmenting the two input types from the raw image captured by the sensors [92].

Parallel to the development of these systems, researchers looked into the theory underlying this particular interaction paradigm. They considered the problem as a special case of asymmetric bimanual coordination, which had been studied for simple two-hand interaction. In his seminal work, Guiard introduced the kinematic chain model, according to which the non-dominant hand (NDH) sets the frame of reference for the dominant hand's (DH) operations [82]. In this model, therefore, the DH and NDH have well-assigned roles when performing bimanual activities. In a modern pen and touch context, this translates as touch performing coarse manipulations, such as panning, zooming etc. while the pen is used for inking and finer-grained actions such as lasso selections and precision gestures. But the simultaneous availability of the two modalities goes beyond providing the possibility for the two hands to perform their specific work without tedious context switches. As shown by Brandl et al. and Hinckley et al., it also allows for richer interaction capabilities, when the two modalities are used synergistically, especially, when the NDH is used to modify the operational context of the pen-holding DH [44, 93]. For instance, a flat hand on the surface can constrain the pen to draw straight lines, or holding an object down with a finger of the NDH while the pen performs a dragging gesture away from it can be used to create a copy of the object [92].

This faculty of pen and touch to be used together in order to produce different functions or actions has been exploited in a few research prototypes already. The tasks and scenarios considered so far include sketching [92], painting [44], diagram editing [75], laying out widgets [77], equation writing [213], 3D modelling [131] and real-time strategy games [85]. Those projects provide examples of how relatively complex activities can be efficiently performed using the two modalities working at times separately in their respective functional scope, but also together to form gestural phrases or "cognitive chunks" [54], especially when pen and touch are combined to build compound commands. Those sequences of unitary inputs and multimodal actions create a fluidity and a continuity that contribute to increasing work efficiency while, at the same time, reducing the cognitive load imposed on users by reducing the need for context switches [93].

1.3 Motivation

As stated above, there are currently very few dedicated document applications for digital tabletop systems that can rival the breadth and comprehensiveness of their counterparts for desktop computers. Our ability to perform productivity work with multitouch devices as well as with digital pens is still largely untapped and the promise of the digital workdesk (and more broadly that of the office of the future) remains to be fulfilled. This work's primary goal is to partially fill that void in the area of document engineering, in particular with respect to the tasks of document editing and composition. It aims to lay down the foundations, or at least propose a path, to inform the design of future pen and touch document production systems. This path need not necessarily share much with that of mobile devices such as tablets, as the interactive contexts of large, fixed digital workdesks and smaller tablets differ markedly. Those discrepancies are all the more apparent in current mobile office suites, which suffer from characteristic problems: as with most mobile applications that have original desktop counterparts, the former usually come with only a subset of the features of the latter. Furthermore, most mobile systems require applications to run in full-screen mode, making it cumbersome to transfer content from one program to another [112]. As document composition often requires referring to other sources and programs, this is a major impediment. In the case of Office 2013, critics have pointed out that efforts to adapt the application UIs for tactile devices have been very minimal, as they have mainly consisted in including a touch on/off toggle to space out widgets in the ribbon interface for more precise finger access. Much of the interface design is based on its WIMP legacy [47]. Generally, one can observe that the majority of touch applications today rely on simple interaction patterns: single finger tap for selecting, drag for panning, pinch/spread for zooming etc. The use of such a reduced set of elementary interaction techniques is prescribed by platform vendors in application design guidelines put out for developers and gestures deemed even slightly more "complex" are discouraged (e.g. Microsoft finds that a double tap is too difficult to time for users and so should be avoided [21]). Advocating a simplified interaction model may be understandable for mobile touch devices with limited interaction space, but seems overly restrictive for larger surfaces such as tabletops. As researchers have shown, multitouch has much more to offer in terms of interaction possibilities [76, 200, 207, 208], to say nothing of pen and touch.

Finally, a problem that plagues all current touch devices is the matter of low text entry efficiency. Text input productivity on that kind of appliances is reduced because of cramped on-screen keyboards and the absence of adequate navigation keys to quickly move the cursor around. While some of the latter problems can arguably be attributed to early design flaws that will be remedied over time, there are more fundamental limitations hindering smaller devices resulting from the sacrifices they have to make for the sake of mobility. Large interactive surfaces, on the other hand, do not have most of those handicaps, while they retain all of their advantages (except mobility, of course), which is why they are attractive platforms for many of these tasks.

Within a pen and touch document editing context, an essential question that requires thorough consideration is the presentational and interactional aspects that the tabletop UI should exhibit. A professional desktop publishing program includes numerous functions, from very basic tools for content insertion and editing, to layout design, advanced styling, support for layers and multimedia elements, publishing options etc. While this work certainly has no ambition to produce a full-fledged product-quality solution, core UI design issues need to be addressed to demonstrate how an efficient pen and touch document editor with basic features could look. In essence, the question to be answered in that particular context is "how should a tabletop publishing application be designed"? This interrogation entails many others. When dealing with NUIs especially, decisions need to be made about the approach to take regarding how that interface will expose its functionality to the user. Should the interface rely on visible widgets that continuously reveal its tools through menus, buttons and other widgets? Or should the application commands, on the contrary, remain unobtrusive and be triggered by gestures? Is it in fact possible to achieve the latter in the context of a relatively advanced tabletop editing system? The pen and touch input paradigm certainly allows for a richer interaction vocabulary and the advantages of having a clean, uncluttered interface are obvious, but is there not a risk that users will reject a system that feels somewhat arcane because they do not have an immediate view of its controls (or lacks sufficient "signifiers", to borrow Norman's terminology [146])? The motivation to provide a clean interface "as is" often arises from another concern that many NUI designers have, that is, how far the virtual system should mimic real interactions in the physical world and hence to what extent it can avoid artificial intermediaries such as widgets. For document tasks, this usually translates to how far the interface should follow a paper/workdesk metaphor, where virtual documents are manipulated in similar ways to their real counterparts. If maximising intuitiveness is the goal, then designers should strive to create interfaces that attempt to faithfully recreate the interactional conditions of the real world, for example by modelling documents as elements that can be scattered or piled on a surface and whose pages can be flipped. While this may increase the initial accessibility of the system for a majority of people, it might later prove cumbersome and unwieldy as its interaction model may turn out to be inefficient for particular tasks (e.g. non-sequential content navigation, custom presentations of information breaking with the original layout). Microsoft Bob[5] and Magic Cap[6] are two commercial examples, where the metaphor was pushed too far and sacrificed too much usability for the sake of recreating a real office environment. At the other extreme, if only practical concerns dictate design choices, then even if the interface promises high effectiveness in the long run, the costs of a steep learnability curve might deter most people. The design of the interface should therefore likely strike a compromise between naturalness and ultimate efficiency. Not only is it a challenge to find such a balance, but once a solution is proposed, it is difficult to evaluate it, considering the diversity of potential users, who each have different

[5] http://en.wikipedia.org/wiki/Magic_Cap
[6] http://en.wikipedia.org/wiki/Microsoft_Bob

habits, experiences and preferences. Moreover, it is difficult to assess the pertinence of design choices for specific functions, when users are given the whole application to test.

As was briefly touched upon above, text input is a crucial aspect that still hampers the wide adoption of NUIs as document authoring systems. For document editing especially, text entry needs to be an effective and pain-free process. On the desktop PC, the keyboard has established itself as the most common tool to perform this task. With practice and different layout optimisations, typists can easily achieve performances above 50 words per minute (wpm), experts even above 100 wpm [32]. Drawing upon this legacy, touch-based devices typically provide on-screen or virtual keyboards to enter text. Due to the lack of tactile feedback, typing speeds are typically lower than for physical keyboards, although most studies have only compared mobile devices [96]. Virtual keyboards rely on people's familiarity with their physical counterparts, but there are also other popular input methods such as stroke-based text entry (or shape writing), handwriting (with recognition) and speech [133, 134]. For the most part, those alternative techniques have been implemented and evaluated on mobile devices such as smartphones and tablets and there is no real comprehensive assessment yet of how well they fare on tabletop platforms.

Despite engineers' best efforts, it is very likely that text input on keyboardless systems will remain less efficient than on a traditional desktop PC, at least in the short term. A question that therefore naturally arises is whether pen and touch tabletop systems should not be geared for specific document types and tasks. Suitable scenarios would play to the platform's strengths and avoid its weaknesses. For example, documents with a strong reliance on pen-drawn input (e.g. sketches, plans and forms) seem to be more adequate candidates than text-intensive documents such as articles, reports, novels etc. Similarly, documents compiled from several pieces of external material can stand to benefit from an interactive pen and touch environment. In such a context, users could rapidly assemble new documents from existing elements extracted from other documents or from a clipart database. Those elements could also be effectively laid out and aligned using appropriate multitouch gestures, as shown by prior work [76, 200].

In a broader perspective, one might also consider more complex workflows that involve multiple platforms, where each is used for a dedicated task. In many industries that involve the creation and publishing of content, the division of tasks according to different roles and specialities already exists, yet most of the work is carried out on desktop computers and only the software differs (e.g. desktop publishing software for layout editors and designers, word processors for reporters and contributors, copy editing tools for sub-editors, proofreaders etc.) With the emergence of new touch-driven devices, including digital tabletops, specialisation can be envisaged not only in terms of software, but also hardware. For instance, the general layout of a publication could be designed on a pen and touch tabletop, the content redacted on a regular computer (or even on a tablet for on-site reporting) and its arrangement on the page(s) performed again on the tabletop. Such workflows may appear more complex and require an even tighter integration of the processes involved to function, but at the end of the day, they may yield substantial productivity benefits if the most adequate tools are used at each step of the chain.

Before embarking on application prototype development, it is worth examining the yet understudied properties of the pen and touch input paradigm. Among essential questions that do not yet have definite answers are the roles of pen vs. touch and how the two modalities should be used together in a document-centric application such as an editor. The division of labour principle strongly suggests that the NDH should only perform operational manipulations such as function activations and context setting, while the pen-holding DH is assigned to inking and the execution of precise actions, such as lasso selections, stroke gestures etc. However, some studies hint that this might be overly restrictive, especially considering that users do not necessarily express clear preferences for one modality when asked to perform a certain type of operation [75]. For instance, it is not clear that given the choice to tap a button either with a finger from their NDH or with the inking pen, most people would choose to use the former. Conversely, we can see that tactile interaction increasingly encroaches on the pen's traditional territory in areas such as drawing and handwriting. The success of the social drawing game Draw Something on multitouch devices is a testimony to that trend [6]. Those aspects will therefore need to be taken into consideration when studying the properties of pen and touch interaction.

Another factor that plays an important role when dealing with two potentially competing modalities is interference. Putting aside low-level issues when pen input needs to be differentiated from touch at the software level, for instance, through segmentation of the raw image data returned by the tabletop sensors, interference in pen and touch systems mostly manifests itself in touch-based actions being accidentally triggered by the writing hand or arm that is resting on the surface. To deal with this issue, the hardware or software has to discard those undesired touch inputs through what is commonly referred to as "palm rejection". Most of the current systems achieve this by simply allowing only one input mode at a time, that is, when the pen is used for interacting with the device, touch is completely deactivated and vice versa. Such a solution is relatively easy to implement, but of course it precludes simultaneous pen and touch interaction. More flexible techniques consider the size of the contact areas (pen and fingers have small diameters compared to arms and palms) or whether touch points are registered in the vicinity of the pen [24]. Beyond the technical solutions required to tackle the problem and the fatigue aspect when using the pen with a raised arm for a long period of time, it is interesting to find out how much of an impact palm resting has on the output of the actual task, especially in terms of precision.

Finally, the question of asymmetric bimanual coordination merits special attention, as it is at the heart of the "pen + touch = new tools" proposition. How best can the two modalities be combined to fully leverage the potential of simultaneous pen and touch input without taxing users' concentration? Prior work offers some intuition as to what kinds of bimanual gestures can be performed in this way. Following the kinematic chain model, pen and touch gestures are achieved through the combination of atomic actions, with touch mostly used for holding objects, setting constraints or triggering a certain mode while the pen inks. In other words, a static NDH sets the context of the moving DH, which seems sensible and not mentally demanding. Pen-mode switching by the NDH, especially, is a powerful asset of the pen and touch input paradigm as it allows users to change the function of the pen with minimal coordination effort. Where in unimodal pen contexts a tracing action with the stylus would have to be interrupted to perform a selection or a widget operation, with interfaces based on two-handed input, this task can be delegated to the NDH so that time-consuming context switches can be avoided. In the extreme, one could conceive of on-the-fly pen-mode changes, i.e. modifications of the pen's operational context during a stroking motion. How far this can be achieved without causing errors is yet undetermined, as currently no thorough empirical study investigating to what extent the coordination effort between the pen-holding DH and NDH can be minimised exists. This question along with the previously mentioned aspects merit, therefore, further study.

1.4 Contributions

The main thesis of this dissertation is:

Post-WIMP interaction and UI designs taking advantage of the affordances of digital tabletops and hybrid pen and touch input allow the creation of intuitive and efficient applications for productivity document engineering work such as document editing and authoring.

Towards showing that and addressing the questions raised above, this work follows a loose bottom-up approach, weaving through a path starting with software design aspects of pen and touch-based applications, followed by an investigation of low-level properties of the input paradigm and moving gradually up towards application-level concerns related to document engineering tasks, in particular document creation and editing. Naturally, this exploration takes place within the chosen setting of interactive tabletops, i.e. large horizontal surfaces at which users sit to work.

In a first step, a software architecture aimed at facilitating the implementation of pen and touch applications is described. Although mainly designed with a view to support prototype development within the scope of this project, several of the concepts are generalisable and applicable in other contexts. The framework differentiates itself from other endeavours in several aspects: it defines a unified touch input model that takes into account the constraints of the DiamondTouch (unlike other frameworks relying on TUIO-style protocols that assume touch contacts are points or ellipses), it provides explicit support for posture- or gesture-based pen-mode setting and bimodal gestures with integrated recognition engines, and finally it includes a set of flexible pen and touch components with built-in behaviours for clients to use and extend. At a later stage, a data-management layer to handle transparent object persistence is added to the framework to fulfil the needs of the second prototype.

In a second part, intuitions and hypotheses posited about pen and touch interaction in prior work are examined. Those aspects include the division of labour principle and role assignments for the DH and the NDH as well as proposed uni- and bimanual gestures. Empirical validation (or invalidation) of those considerations is provided through a series of trials performed in a controlled user study. Specifically, the two modalities are tested in terms of precision and suitability for executing basic tasks and operations, such as widget targeting and positioning, shape tracing and tool selection. Bimanual coordination is also studied through a set of different pen-mode switching actions triggered by the NDH. Particular contributions, here, are measurements of the displacement-on-release effect for pen and finger (that is, the amount the contact point moves upon releasing the pointer), the impact of palm resting on tracing precision and performance assessments of maintained NDH postures (as an example of a quasimode [160]) to control pen strokes on-the-fly. The experiments also reveal valuable insights into users' modality preferences for tracing and selection tasks.

Thirdly, a first document-centric application is developed to support the activity of active reading (the task of reading a document, not necessarily sequentially, while also taking notes and referring to other material). The prototype proposes an alternate approach to other AR applications described in the literature. Through its pen and touch techniques, many of which are modelled on real-world pen interaction paradigms (the UI also follows a codex metaphor with documents, whose pages can be virtually "flipped"), the trade-off between realism and pragmatism for interaction design is examined. Novel non-inking techniques inspired by physical pen-on-paper interactions include multiple page flicking and dynamic side bookmarks containing search results and colour-coded hyperlinks to annotations. The former reality-based multiple-page-viewing method contrasts with the other integrated technique: a chop-and-spread gesture that transitions in a more computer-style way to an overview screen. Blending real interactions and computer tools are hybrid commands constructed around quasimodal (i.e. maintained) activation of functions, whose parameters are expressed by the pen, e.g. handwritten search keywords and pages to turn to. The user evaluation provides answers as to how effectively the tabletop system is able to combine the advantages of physical paper (intuitive and immediate interaction) and that of a traditional computer platform (ability to efficiently search for content), without their disadvantages. The results show that this is largely the case, but that the desk/codex metaphor should not be pushed too far for the sake of keeping users in a familiar territory. Other important findings of this study deal with hardware limitations and the extent to which they affect users' experience. Feedback provided by participants brings to light the importance of the viewing resolution for readability as well as the responsiveness, which prove to be particularly crucial in those kinds of document tasks.

The second application focusing on multi-document authoring is the main contribution of this thesis. Within this effort, several gesture-based implementations of common document operations are presented, together forming a feature-complete coherent whole. In particular, the concept of touch-modified pen stroke gestures is applied more systematically with NDH-postures defining specific categories of pen transactions. The resulting commands are likened to keyboard shortcuts, which similarly combine modifiers with regular keys that normally input data when activated on their own, but take on a new role when modified. Interactions consist of unimodal multitouch gestures mainly for document navigation and bimodal bimanual techniques to execute a variety of commands including page creation, text and shape styling, undo/redo and content registration/insertion. This reliance on gestures and pen and touch operations allows the interface to resort only sparingly to UI widgets. The report indicates, however, when explicit helper tools are likely unavoidable, e.g. for font and colour selection. Text entry is also tackled and novel techniques to recreate standard word-processing behaviours using handwritten input and multimodal gestures are proposed. The application further integrates methods to seamlessly retrieve and insert document elements from external sources (such as a clipart database) using query-by-sketching and associated pen gestures. Finally, the lab study provides a first assessment of the pertinence of the approach, with findings ranging from the impact of gesture relaxation on conflicts to estimations of suitable phrasal complexity for sequenced hybrid interactions.

For both document application scenarios, several suggestions for possible enhancements and extensions in future research are given. The role of the pen as a text-entry tool, among other functions, is also briefly examined.

1.5 Outline

The remainder of the thesis is structured as follows: in the next chapter, the background to this work is provided with a review of the related scientific literature. Specifically, the evolution of digitally enhanced workdesks and tabletop interfaces is presented, along with relevant technological developments concerning natural interaction techniques such as multitouch and digital pens. Within the context of document engineering, systems such as interactive paper, tablets and NUIs targeted at document-centric tasks are discussed. In the final part of this chapter, the theory underlying pen and touch interaction as well as the systems based on that input paradigm that have been produced so far are described.

The third chapter introduces the software framework that was created to form the backbone of the pen and touch application prototypes developed for this project.

The fourth chapter deals with the properties of pen and touch input and the empirical verification thereof. The four experiments investigating attributes such as precision, degree of freedom control, inter-modality interference and bimanual coordination are presented, followed by a thorough analysis of the results and the design implications for pen and touch systems.

In the fifth chapter, the steps that punctuated the development of the first prototype application dedicated to active reading are detailed. A report of the user study that compared the tabletop application to two other traditional media (a desktop PC and paper) within the context of three active reading tasks is provided.

In the sixth chapter, the document composer prototype is described, from its structural aspects (i.e. its software building blocks) to its interactional features. In particular, the proposed modifier posture model, upon which many of the gestures are based, is explained. The pertinence of this model, aimed at speeding up common document operations as well as reducing the reliance on UI widgets, is discussed in consideration with the extent of the feature set that a typical document editor requires. The chapter is wrapped up by an observational study, in which users who tested the system provide their feedback and criticism.

Finally, the concluding chapter sums up the thesis and its contributions before offering a few avenues to explore for future work.

2 Background

The present work was preceded by a considerable amount of achievements both in the research community and in the industry sector. The main highlights of relevant literature and prior systems are presented hereafter, starting with a brief review of the concept of the office of the future and its goals. This introduction is followed by an examination of interactive tabletops and workdesks, pen-based document systems, including tablets and interactive paper, the pen and touch input paradigm and its applications so far. The chapter concludes with an analysis of relevant issues that emerge from the reviewed prior art.

The subsequent chapters will also cover some more related work dealing specifically with the material introduced in them (in particular, user evaluations and studies).

2.1 The Vision of the Office of the Future

For many decades, the organisation and modernisation of the workplace has occupied the minds of industrial designers and practitioners dreaming up visions of the office of the future. One of the stated goals at the time was to achieve a "paperless office", i.e. an efficient work environment in which paperwork is replaced by entirely electronic transactions. Popular images for such concepts are rooms filled with displays, interactive surfaces and sensors that are able to show and capture information in real-time. Those rooms can also be virtually extended to include remote participants blended in to appear locally in 2D or also sometimes in 3D. In our current era of global mobility and interconnectivity, the concept of a single common location in which all work is done (the workplace) has now been superseded by a more nuanced picture of increasingly nomadic workers, who are able to perform their tasks from virtually everywhere with a variety of smart devices. Furthermore, information that used to be coarsely harvested and pushed to the user (necessitating a great deal of filtering) in the past is now more intelligently processed so that systems are able to know when and how to present it to the user, thanks to their enhanced context-awareness.

In terms of user-computer interaction, natural input methods such as touch, pen, 3D gestures and speech are foreseen to become even more prevalent, both for small and large devices. Because almost every surface is interactive and interconnected, content can be easily transferred from one appliance to another to be manipulated or shared with other people. The digital workdesk itself becomes a fully interactive space on which workers can view relevant information as well as control objects appearing on their front display via smart on-screen tools or speech commands.

Many visions about future work practices and how technology will support them have been reified in concept videos. One remembers Apple's famed Knowledge Navigator described in former Apple CEO John Sculley's 1987 book Odyssey [171] and showcased in a number of short films created by the company. Those videos depict digital tablets as personal devices with pen and speech-based interactive capabilities, many of which the company has integrated in its current consumer products. More specifically concerning the office of the future is Tognazzini's Starfire video prototype project conducted at Sun Microsystems in 1994 [190] (Figure 2.1 left). The film features a large display consisting of a curved vertical screen and a flat desktop surface. The user interacts with the device by voice, mouse, stylus and gestures to perform a variety of tasks, including editing multimedia documents. More recently, Microsoft produced two videos painting its vision about how people could be working in the next decade [12] (Figure 2.1 right). Those films give a more prominent role to mobile devices and underline how they will become an even more integral part of employees' daily lives. The digital office desk is still present, but in many aspects, it looks like an updated version of the display featured in Starfire with similar interaction patterns. It is incidentally interesting to observe that, in the early concept movies, the keyboard was notably absent, as text and commands were input mostly through speech and to a lesser extent with a stylus. Microsoft's 2011 video, however, reinstates the keyboard and projects it will remain one of the main instruments for text entry in the future.

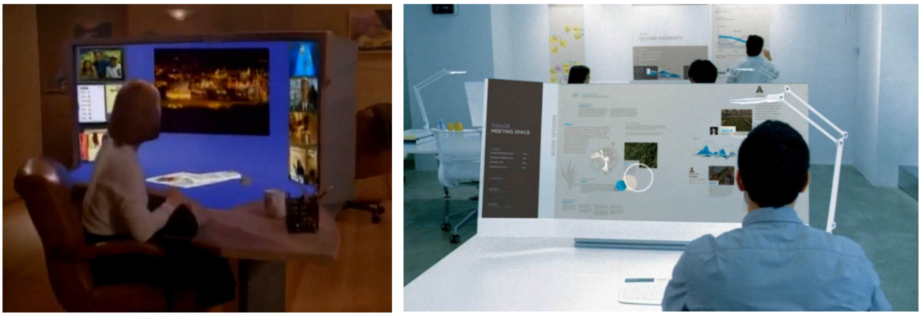


Figure 2.1: Future digital desks as imagined by Sun Microsystems in 1994 [190] (left) and Microsoft in 2011 [12] (right).

The vignettes of futuristic UIs that appear in concept videos and science fiction films may not exist in the real world, but several ideas have already been envisioned by researchers and materialised in actual prototypes. In 1998, Raskar et al. proposed and partially implemented an immersive office environment with an array of projectors and cameras creating virtual and augmented spaces to display and interact with information and people beyond the small screen of a desktop computer [159]. In particular, the authors were motivated to support telecollaboration by virtually merging two remote offices in a single one (Figure 2.2). Their work highlights the technical challenges of developing advanced computer vision algorithms that are able to understand the spatial context and compute realistic projections of objects or people on uneven surfaces.

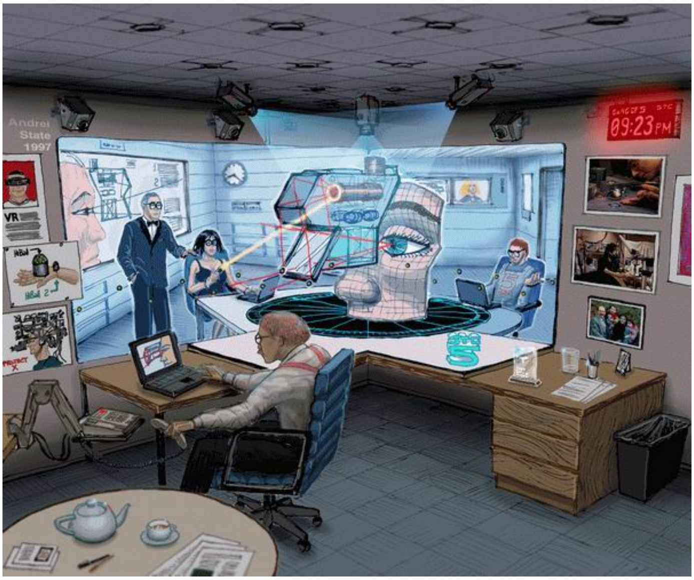


Figure 2.2: Conceptual sketch of Raskar et al's Office of the Future involving multiple cameras and projectors fixed onto the ceiling [159].

A further embodiment of interconnected devices, room elements as well as furniture items was proposed by Streitz et al. who encapsulated the concept in the term Roomware [181]. Their approach envisaged the extension of the traditional office environment and its components to integrate the circulation of information and communication technology even more tightly. They viewed these components as constituents of so-called Cooperative Buildings [180]. Within the course of their project, they designed several Roomware components, including the DynaWall (an interactive wall), InteracTable (an interactive table), CommChair (mobile and networked chairs with integrated interactive devices) and ConnecTables (a pulpit with touchscreen that can be combined with other ConnecTables in order to create a large display). Their contribution also included a software infrastructure to manage those components called BEACH. This framework provided UI and interaction support as well, including gestures to orient content for multiple users.

Another solution that contemplates hybrid workspaces mixing computers, tables and walls was proposed by Rekimoto and Saitoh [162]. The researchers created an augmented environment in which the surroundings function as spatially continuous extensions of personal computer displays so that users are able to move content to and from those ad hoc surfaces (a notion they called "hyperdragging"). The work also describes a mechanism to attach digital data to physical objects.

The projects described above, initiated at the turn of the millennium, have in common that they consider the office space and work environment as a whole. They present views of augmented workspaces in which information can be visualised and shared with little restriction, thanks to smart networked office equipment. Within this modern context-aware paraphernalia, the interactive workdesk plays an essential role. As a potential substitute of the regular PC or at least a valuable supplementing tool, it stands to provide increased and more intelligent support to knowledge workers in the execution of their tasks.

2.2 Interactive Tabletops and Workdesks

Digital tables and interactive surfaces, as part of the family of large interactive spaces have a history that goes back to the 40's with the pioneering Memex, an early concept of an augmented desk proposed by Bush in 1945 [51]. The engineer envisioned a machine capable of compressing, storing and visualising books that "may be consulted with exceeding speed and flexibility". Actual prototypes of interactive desks were only produced in the early 90's at Xerox EuroPARC with Wellner's influential Digital Desk [198]. In the following 20 years marked by considerable technological improvements, systems evolved from bulky and cumbersome setups, involving a great deal of equipment above and around the working surface, to radically smaller and leaner machines integrating all the necessary sensing technology inside them. Along with this reduction in size, tactile appliances have also made great strides in terms of sensing resolution, responsiveness and accuracy.

While many technological similarities exist between horizontal and vertical interactive surfaces, one should carefully distinguish the two categories, as their interaction patterns differ quite significantly. Wall screens and whiteboards are often used for activities such as brainstorming, drawing rough lists, mind maps and presenting information, whereas tables and desks are typically platforms on which more elaborate work is carried out, at least in single user situations. In multi-user settings, common scenarios range from casual interactions such as sharing photos etc. to meetings and collaborative work. In those situations participants find themselves sitting or standing around the table. This kind of configuration poses a number of problems, including content orientation, territoriality, reach, user interference and possible occlusions through physical objects placed on the surface. Those issues are extensively studied in the discipline of CSCW and are outside the scope of this thesis.

There follows a brief chronological overview of the major tabletop systems that have been built to date. More detailed and complete listings are available in [53] and [115].

2.2.1 Historical Interactive Tabletops

The Digital Desk

The first breakthrough in augmenting a physical workdesk with digital interaction facilities was achieved in 1991 when Wellner invented the Digital Desk at Xerox PARC [198]. His system consisted of a regular office desk surmounted by overhead projecting and sensing equipment. While the projector beamed images onto the desk, the camera tracked finger movements as well as recognised and captured paper documents placed on the surface. Touch contacts of fingers were detected using a microphone attached to the bottom of the desk. A high resolution camera was available to acquire document content such as text (then decoded by an OCR system). This setup allowed users to interact seamlessly with a blend of electronic (the projected image) and analogue material (the paper documents). Wellner's example application for his prototype was a calculator, where numbers appearing in paper documents could be directly pointed at with the finger to input them in the system.

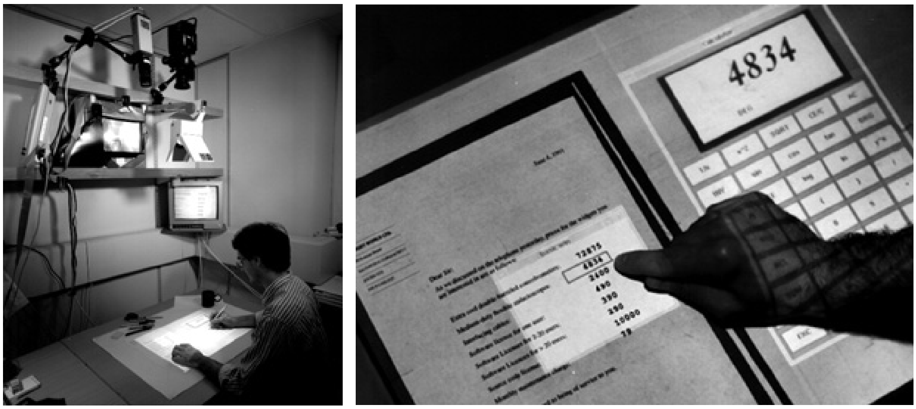


Figure 2.3: Wellner's Digital Desk with the calculator application

The Active Desk

The Active Desk, whose first version was created in 1992 at the University of Toronto was an electronic drafting table controlled by a stylus to support drawing activities [52]. In 1995, Fitzmaurice et al. extended this platform with physical control artefacts, or "bricks" to operate associated virtual objects, introducing the concept of "graspable user interfaces" [71]. An unpublished project involving an Active Desk and an overhead camera to track a grasping gesture of the non-dominant hand allowed simultaneous bimanual stylus and touch interaction, albeit with a somewhat reduced touch gesture vocabulary [52].

The MetaDESK

The metaDESK developed in 1997 by Ullmer and Ishii was meant as a platform to explore the design of tangible user interfaces, that is, the possibility to interact with computer-generated objects using everyday physical objects [192]. In terms of hardware, the system consisted of a back-projected desk, a set of lenses (an arm-mounted LCD screen to view 3D scenes and a passive glass lens that could be placed directly on the surface to view an additional layer of information) and an array of optical, mechanical, and electromagnetic field sensors to detect objects. This seminal work led to Ishii's more general Tangible Bits vision, aimed at promoting the exploitation of human's natural ability to interact with objects in the physical world and apply those principles to the control of digital information [98].

The DiamondTouch

Invented in 2001 by Dietz and Leigh at MERL, the DiamondTouch is a front-projected touchscreen that relies on an electric current forming a loop through a transmitter in the table and running through the user’s body [65]. A circuit is closed when the user touches the surface, causing the system to detect a contact with the surface. By transmitting different electrical signals through its wires, the hardware is able to precisely identify different users interacting at the same time with the table. This ability made the DiamondTouch a popular device among researchers for group applications and experiments in collaborative situations.

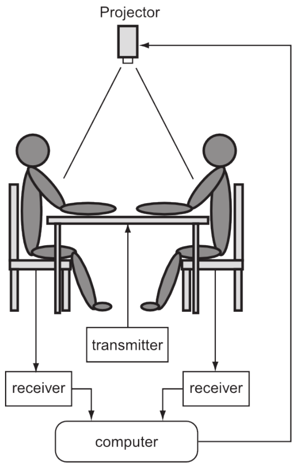


Figure 2.4: Schema of the DiamondTouch's signal loop (left) and a meeting scenario where the table is used (right)

Due to the configuration of the sensors placed in the table bezels, the system is only able to detect signal peaks on each axis corresponding to vertical and horizontal projections of contact areas. A consequence is that the DiamondTouch cannot unambiguously distinguish multiple finger touches from a single user. This is a disadvantage compared to other tabletop devices that receive full 2D input frames from their digitisers to determine contact points. Techniques have been developed to try to infer those contact points using software tracking schemes [38], but they rely on touches not occurring simultaneously, and hence are not perfect workaround solutions. What the DiamondTouch lacks in touch ambiguity, however, it makes up in responsiveness and robustness, as the system proves considerably reactive and reliable compared to vision-based setups, for example.

SmartSkin

Rekimoto's SmartSkin is a tabletop device that uses capacitive sensing and a mesh-shaped antenna inside the apparatus [161]. Through capacitance changes induced by objects approaching the surface, the sensors can roughly measure their distances to the table (this allowed, among other things, the detection of grab gestures to "pick up" virtual objects). Like vision-based systems and unlike the DiamondTouch, SmartSkin is able to detect individual hand and finger positions with a more or less high degree of precision, depending on the density of the electrode layout.

ReacTable

This project started off as a desire to build a novel computer-based musical instrument. The concept came into being as an interactive round table on which special pucks were placed and manipulated to control the generation and modulation of electronic music [101]. The system consisted of a back-projected round table with IR detection of tangibles with so-called fiducial markers, i.e. unique visual tags recognisable by the system's computer vision algorithms. The ability to detect tangibles' orientation as well as finger touches allows for a great degree of interaction and parameter-control possibilities.

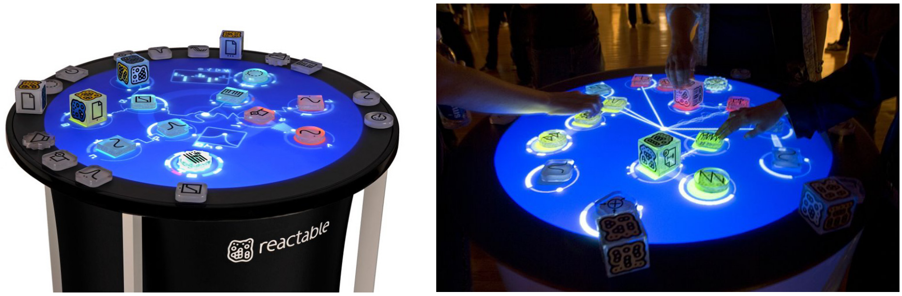


Figure 2.5 The reacTable [101]

ReacTable enjoyed wide international success with numerous presentations and concerts around the globe. In 2009, the inventors created a spin-off company to bring the device to the mass market.

PlayAnywhere

Wilson's PlayAnywhere introduced projecting and sensing from an oblique angle [202]. The device consisted of a projector and calibrated camera that could sit on any flat surface, with which users could freely interact. The touch detection technique was based on recognising shadow shapes produced by an infrared illuminant (which also allowed interaction with physical objects). An optical-flow algorithm was responsible for determining hand movements and on-screen object manipulations.

Perhaps one of the major practical benefits of this system is that it was constructed as a single piece, thereby alleviating problems of calibration, installation and portability that are common to other setups.

FTIR

FTIR (Frustrated Total Internal Reflection) is an optical phenomenon at the basis of the multitouch system created by Han in 2005 [86]. It relies on infrared light trapped inside an acrylic panel by internal reflection and diverted towards a camera when there is a touch contact on the surface. Contrary to most previous systems, the sensing and projecting equipment was placed underneath the tabletop surface and hence there were no occlusion problems resulting from shadows etc. The size of the box made it however less comfortable than a normal table, as there was little legroom for sitting users.

Microsoft PixelSense

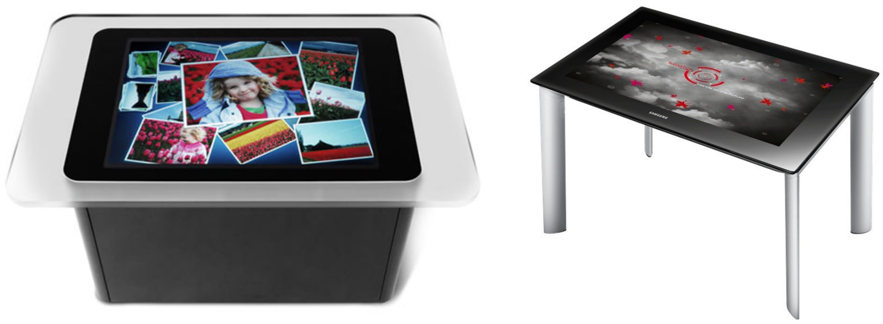


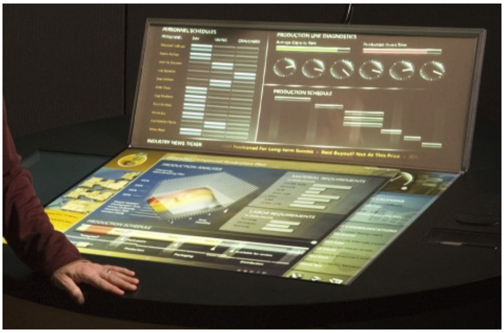
Figure 2.6: PixelSense 1.0 (left) and 2.0 (right)

Microsoft PixelSense (formerly Surface) is perhaps the tabletop system most widely known by the public, due to the relatively strong marketing backing of its parent company. The first Surface was a rear-projected table with five near-infrared cameras that read touches on the surface and an integrated PC for processing the data (and hosting the applications). While Surface had been in the works since at least 2001, it only appeared on the market as an end-to-end solution in 2008 and was sold primarily to businesses. The product shipped with an SDK allowing customers to develop their own applications, including a tag library to automatically recognise fiducials.

Surface 2.0, renamed to PixelSense in mid-2012, is the current version of Microsoft's surface computing platform. The hardware manufactured by Samsung is an LCD panel with embedded sensing technology and computer. The difference to the previous version is that the IR sensors are integrated into the RGB pixels of the screen as a fourth component, i.e. they are directly incorporated in the panel itself, which allows it to have a much thinner form-factor. Thus, the device can be easily deployed either as a table or as an interactive wall screen.

One of the major drawbacks of relying on IR sensors is the sensitivity to ambient light, which can interfere with the detection of touch input. In order to function properly, PixelSense tables need to be operated in relatively dark conditions.

2.2.2 Non-Planar Desks



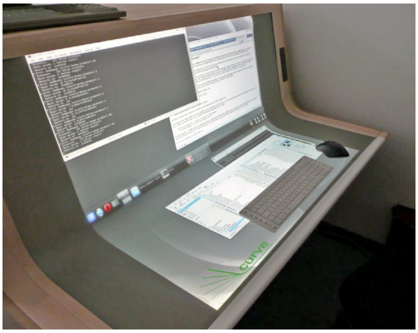


Figure 2.7: Microsoft's DigiDesk workstation, demoed at Convergence 2007 (left) and Curve, a prototype multitouch curved desk by Wimmer et al. [204] (right)

As seen in some of the projects featuring combinations of vertical and horizontal devices (as well as in the office of the future concept videos), there is potentially a need for users to be able to interact with various types of surfaces with different orientations and curvatures, as each have their own advantages. Hence, in such types of contexts, both vertical screens and horizontal workdesks function as displays and interactive surfaces so that there is no passive screen and no proxy manipulation from a separate interactive space. But with such a requirement arises the problem of transitioning between the different surfaces, in particular to move one on-screen object to another display. For relatively distant interactive spaces such as a table and a whiteboard, pick-up or grab gestures can be used [162, 203], however in case of nearby surfaces such as a desk and a monitor this seems overly demanding. An obvious solution to this problem is to combine horizontal and vertical planes in a single non-planar interactive surface.

Perhaps inspired by the Starfire video, in 2007 Microsoft's Center for Information Work produced a prototype of a tilted digital desk with a connected vertical heads-up display. While the screen was not touch-sensitive, it exemplified the appeal of having a continuity between the different surfaces in order to enable smooth, uninterrupted transitions between the different interaction spaces, especially vertical and horizontal planes. This concern was the driving motivation behind projects such as BendDesk [197] and Curve [204] (Figure 2.7), which are curved multitouch surfaces blending horizontal and vertical elements through an arched connecting part. Those prototypes are still in their early development phases, however, and it is not yet clear how UIs should be designed for such hardware profiles and what kind of applications can concretely benefit from them.

2.2.3 Modern Augmented Workdesks

As seen in the previous sections of this chapter, an interactive surface need not be a single monolithic device entirely replacing the worker's office desk. In established work environments with furniture and equipment already in place, it is sometimes preferable to extend or augment the workspace rather than creating a totally new one. This means keeping the user's instruments and objects but endowing the desk surface with interactive capabilities in order to better support the user's work. A practical problem that arises in those hybrid conditions is when physical objects placed on the table such as mugs, books, mobile phones etc. are in the way and occlude part of the displayed virtual content. If the table is able to detect those physical objects (which is not always obvious), the UI engine can rearrange the layout of widgets and controls so that they do not appear below a physical item [100]. Another solution is to allow users to indicate where UI elements should appear, for instance, by letting them draw a path along which menus will be displayed [120].

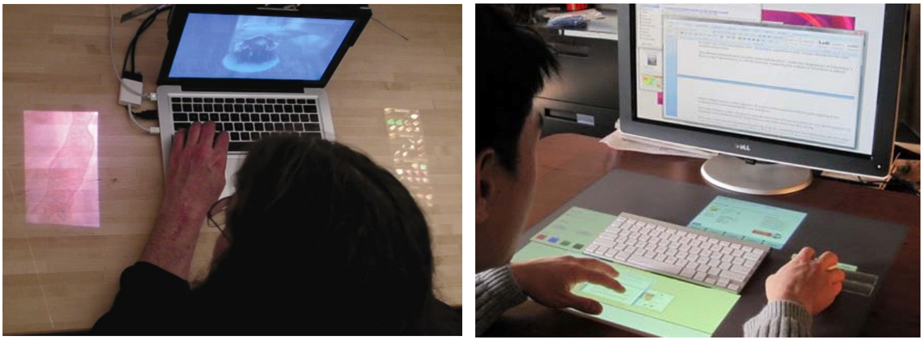


Figure 2.8: The Bonfire [109] (left) and Magic Desk [39] (right) systems

Some other projects consider the specific case of workdesks used as extensions of the interaction space of regular computers (desktops or laptops) placed on them. Bonfire, for example, is a notebook computer augmented with two side micro-cameras and projectors that allow it to project digital content around it and capture peripheral activity, such as the user's hands and objects placed in its field of view [109] (Figure 2.8 left). The device is billed as a "self-contained nomadic system", which affords peripheral display and natural interaction to any surface on which the augmented laptop is placed. Another hybrid system with a particular focus on enhancing the desktop environment is Magic Desk [39] (Figure 2.8 right). Based on a study showing what regions surrounding keyboard and mouse are the most accessible, the researchers define a multitouch space with additional tools such as an enhanced task bar and a digital mouse pad to enrich the interaction possibilities of desktop work.

2.3 Pen-Based Document Interaction

The ability to handwrite is acquired at a young age as part of education and while it is arguably less often used today, due to the omnipresence of computers and keyboards, it is still an essential skill of literate people that can be relied upon. Moreover, handwriting carries a more personal touch that print text does not convey. It is therefore not surprising that engineers and researchers have sought to build systems around this ability and the pen-on-paper experience. Generally speaking, the main goal of those efforts is to take advantage of people's aptitude to use pens and styli and augment that experience with amenities of the digital world, e.g. content linking, persistent storage, efficient retrieval, processing of ink data etc. Here one can basically distinguish two categories of systems: pen-based computers and interactive paper. The former group concerns entirely electronic devices that mimic the affordances of pen and paper such as stylus-operated tablets and interactive tabletops, whereas the latter relates to actual pen and paper contexts but augmented with external capability-adding digital tools. The following subsections elaborate on those two classes of systems.

2.3.1 Pen-Based Computing

A Brief History of Pen-Operated Computers

The origins of pen-based computing date back to the late 19th century with Elisha Gray's 1888 patent describing the Telautograph [80], viewed as the analogue ancestor of the fax machine. The invention consisted of a transmitter, a device that converted pen movements into electrical impulses, and a receiver, which retranscribed those signals into mechanical movements controlling a second pen at the other end. With this machine, it was possible to transmit drawings, handwritten text and especially signatures from one piece of paper to another at a remote location.

One of the earliest pens designed to interact with computer screens was the light pen, created in the early 1950's at MIT and used extensively in the 60's throughout the 70's and into the early 80's. It was notably the input device utilised in Sutherland's famous Sketchpad program [184]. The first electronic tablet with direct pen interaction, Stylator, was built in 1957 and was followed in 1964 by the more popular RAND tablet (also known as Grafacon) [63]. Those systems were already capable of recognising freehand input of letterforms. RAND, in addition, was able to detect basic shapes as well as a scratch-out gesture to delete strokes in a flow chart design application. In 1968, Alan Kay introduced the Dynabook concept, a kind of thin laptop computer targeted at children with one incarnation intended to be operated by a stylus [111]. The ambitious idea, which later inspired notebook computers and tablet PCs, was never turned into a product, although Kay and his team did build a prototype several years later. Subsequent digitisers manufactured in the 70's and 80's include Summagraphics' BitPad, which was a popular input device for high-end CAD applications; the KoalaPad, a low-cost graphics tablet for home consumers that could be used with many personal computer systems of the time, including the Apple II, the Commodore 64 and the IBM PC; GRiDPad, a touchscreen tablet whose progenitor Jeff Hawkins went on to later found Palm Computing and create its flagship device the Palm Pilot; and Wang Laboratories' Freestyle, an office system that enabled workers to annotate documents with handwritten notes on a tablet as well as with voice comments [73]. The late 80's were also the period when the company Wacom released its first electromagnetic induction digitisers, mainly targeted at CAD.

Despite their advertised advantages, many of those early tablet systems were too expensive or too much confined to niche markets to woo customers away from the more popular desktop PC with its keyboard and mouse. The 90's saw the emergence of more affordable and more portable consumer systems, namely EO's Personal Communicator, IBM's ThinkPad (now a family of laptop computers, but the original model, the 700T, was a slate computer) and the first stylus-enabled personal digital assistants (PDAs): Apple's MessagePad (based on Newton) (Figure 2.9 left) and the PalmPilot. To further drive costs down and still deliver a unique experience, vendors developed flexible operating systems tailored to the tablet form factor and designed to be integrated in more than one device, e.g. PenPoint OS, Newton OS and Windows for Pen Computing. In the late 1990's a number of tablet prototypes (NewsPad, Cyrix WebPad, QBE, Intel's Web Tablet) were demonstrated at technology fairs, some of which were later commercialised.

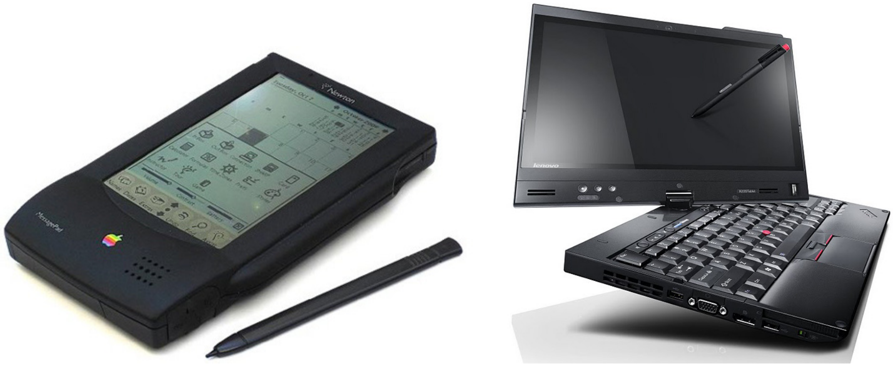


Figure 2.9: The Apple MessagePad (left) and a convertible Lenovo Tablet PC (right)

In 2001, Bill Gates announced the specifications of the Microsoft Tablet PC platform, which comprised particular hardware requirements and the adoption of a special Tablet PC Edition of Windows XP as its operating system (which superseded its earlier pen computing operating environment Windows for Pen Computing 2.0). The first tablet PCs based on those specifications were released by original equipment manufacturers (OEMs) in 2002. Those include full slate devices but also convertibles, i.e. laptop computers with a keyboard and a rotatable touchscreen that could be operated with a stylus. Two years before, Microsoft introduced its Pocket PC specifications for palm-sized devices, i.e. PDAs, which later became part of the broader Windows Mobile platform. The latter was based on Windows CE components, i.e. not a regular desktop Windows and hence required special adapted software.

Tablet PCs and PDAs enjoyed only limited success in the mass market, due to reasons of price, still relatively bulky hardware and, especially for tablet PCs, insufficient software specifically designed for those systems. In 2007, Apple released the iPhone and three years later the iPad, supported by a dedicated operating system specifically designed for multitouch input: iOS. Thanks to their outstanding industrial design and smooth interfaces, those products were able to impose multitouch as the de facto interaction method for smartphones and tablets. Recently, however, there has been a resurgence of the stylus pen, which now comes as a supplementing instrument to multitouch instead of the main interaction tool. Moreover, systems have evolved from only allowing alternate use of touch or pen input to fully multimodal devices, where both interaction techniques can be used together (e.g. the Wacom Cintiq 24HD). Pen and touch systems are analysed in more detail further below.

Pen-Based Document Applications

One of the main foci of document software developed for pen-controlled devices has naturally been to exploit typical pen-on-paper interactions and scenarios. For instance, activities such as drawing and note-taking have received considerable attention from developers and researchers. Annotations made to a document viewed on a tablet can indeed not only be stored, indexed and retrieved at leisure, but they can also be processed and interpreted by the machine to provide an enriched experience to the user. Two seminal (and related) projects from Xerox's FX Palo Alto labs in the late 90's demonstrated the potential added-value that could be obtained from a tablet-based note-taking application: Dynomite [201] and XLibris [168]. The first work laid the foundations of a personal ink and audio notebook. It featured a paper-like interface and functionality to group and categorise the notes for later retrieval. The second prototype, XLibris, was a system aimed at supporting "active reading", i.e. reading while making notes for future reference. The application deployed on a high-resolution pen tablet also allowed users to mark, filter and index freeform ink notes, but additionally, it was able to search for related information and display links to that material on the interface.

The desire to make the most out of freehand pen strokes to interpret and synthesise the conveyed information poses a number of arduous technical challenges, the main of which, perhaps, is accurately recognising and interpreting handwritten text. Since the early days of OCR and pattern recognition, much progress has been made, so that nowadays handwriting recognition packages have achieved very acceptable levels of accuracy and performance (at least, in on-line mode). The high performance of handwritten text recognition algorithms are attested in the results of the competitions regularly held as part of the biennial International Conferences on Document Analysis and Recognition (ICDAR). Depending on the type of the target document, other kinds of recognition algorithms are required, e.g. shape recognition for diagrams and chemical graphics, recognition of equations and mathematical symbols for formulas, recognition of scores for sheet music etc. A full review of those techniques is beyond the scope of this thesis and the reader is referred to the field literature for further detail.

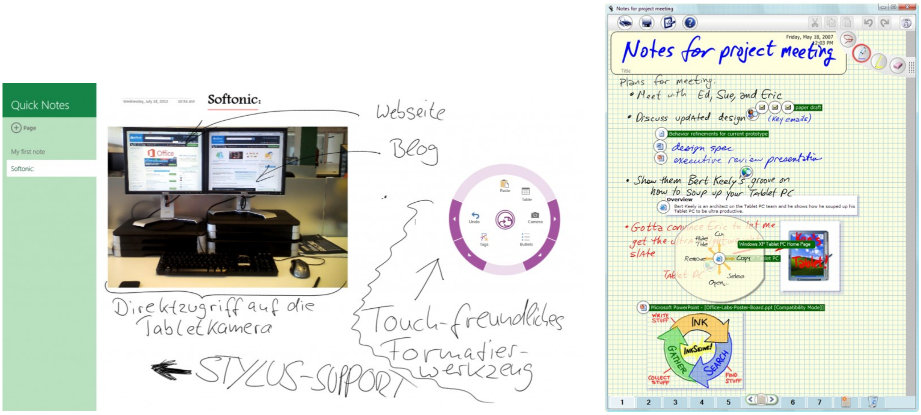

Quick Notes
Page
My first note
Softonic
Softonic:
Webseite
Blog
Paste
Table
Undo
Camera
Tags
Bullets
Direktzugriff auf die Tabletkamera
Touch-freundliches Formatier-werkzeug
STYLUS-SUPPORT
Notes for project meeting
Friday, May 18, 2007
Plans for meeting:
• Meet with Ed, Sue, and Eric
• Discuss updated design
paper draft
(key emails)
Behavior refinements for current prototype
design spec
executive review presentation
• Show them Bert Keely's groove on how to soup up your Tablet PC
Overview
• Gotta convince Eric to let me get the ultra portable slate
Windows XP Tablet PC Home Page
Cut
Copy
Remove
Select
Open...
Microsoft PowerPoint - [Office-Labs-Poster-Board.ppt [Compatibility Mode]]
WRITE STUFF
INK
GATHER
INKSEINE!
SEARCH
COLLECT STUFF
FIND STUFF

Figure 2.10: OneNote 2013 [14] (left) and the InkSeine [94] prototype (right) with pen-friendly radial menus for function selection

Turning to more feature-rich pen-based document applications that go beyond simple note-taking and smart recognition of specialised content, one observes that only very little software exists in this category. The most prominent commercial product specifically designed with those platforms in mind is perhaps Microsoft's OneNote [14] (Figure 2.10 left), which belongs to the Office suite and is incidentally also a note-taking tool, but with a more extensive set of functions. The latest version of the application notably features a "touch friendly" radial menu with multiple hierarchies, from which commands can be accessed. An interesting feature of the menu is that operations can still be triggered even in its collapsed state by performing touch or pen strokes towards the direction in which the corresponding icon would appear if the menu were expanded. In terms of functionality, OneNote supports the integration of screen clippings and material gathered from other applications, a built-in OCR, advanced searching functions and sharing possibilities. Some of those features can also be found in Microsoft Research's InkSeine prototype [94] (Figure 2.10 right). InkSeine's user interface similarly mimics a virtual notebook page on which freeform notes can be made, but its functionality is much less obtrusive. For instance, radial menus only appear when summoned by a pen action. Content from external sources can be extracted with lasso selections and directly dragged into the notebook page or alternatively linked to via a small icon placed near the note item. One of the program's most powerful features is its integrated search function, where keywords can be simply selected from the handwritten text. The search tool allows the user to easily find E-Mails, documents, web pages and other notes in situ, i.e. without resorting to a different interface or a separate widget.

An earlier project dealing with note-processing but whose purpose is more focused on the authoring of documents is SketchPoint [124]. This program is a tool which aims to help users create informal yet structured presentations from handwritten notes and freeform sketches. The document-engineering process in SketchPoint is driven by user-guided mechanisms to infer content hierarchy for the semi-automatic generation of slides. Users control the conversion process through a set of gestures (with, incidentally no mode-differentiation with inking), to indicate, for instance, which items belong together and where hyperlinks should be inserted to jump between slides. The application also features limited segmentation and recognition capabilities for structured content such as text and flowcharts.

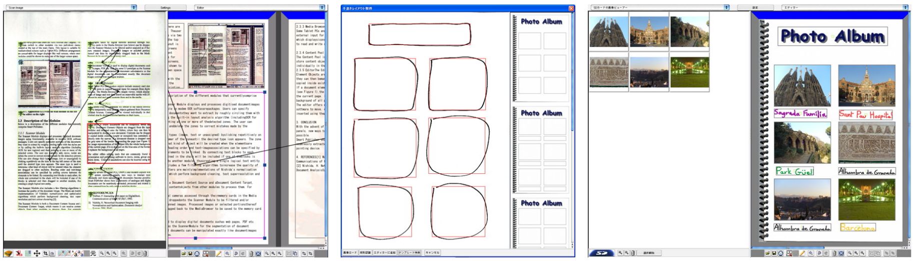


Figure 2.11: The SmartPublisher application. Left: two-pane user interface with the scanner module on the left and the editor on the right. Directional links between text frames indicate flow order. Middle: the layout search and template matching module. Templates most similar to the query are displayed in the right column. Right: content inserted into the placeholders of a photo album template.

This convenience of pen interaction to support rapid gathering of content from various information sources and pasting it into a new document is at the heart of another application with a distinct focus on document creation: SmartPublisher [139] (Figure 2.11). The program, designed for high-end multifunction printers with integrated touchscreens, consists of a two-panel interface: a document page for composition on one side and content extraction modules on the other. With the latter, users are able to seamlessly select elements from different document sources, analogue (e.g. document images obtained from the scanner or a camera) and digital (e.g. web pages) with the stylus and then simply drag and drop their content in the former to compose new documents. The application also integrates a feature to design layouts (a grid of empty frames with page embellishments) and search for system templates based on hand-drawn sketches [138]. Document elements can then be directly inserted into the placeholders, where text content can be flowed from one frame to another using directional links connecting the boxes.

Interactive Paper

Despite technological improvements and the promise of a paper-like experience, digital tabletops, tablets and ebook readers are currently unable to fully match the affordances of real paper. But paper also lacks many conveniences of computers, hence a great interest among researchers to attempt to bridge this gap, by augmenting pen on paper interaction with digital tools. A further and oft cited justification for taking paper as the primary interaction medium is the Myth of the Paperless Office [172], an observation that predictions of all-digital office environments, in which paperwork had been entirely eliminated, had missed the mark, as paper use was not even decreasing but actually increasing. Even though the book was published more than a decade ago and the situation might have changed since (there were no iPads and Kindles in 2001), paper is still very far from extinction in offices and paper is *per se* a document, so projects dealing with interactive paper merit consideration. Besides, many of the interactive techniques and gestures developed for augmented paper applications can also be applied to tabletop interfaces.

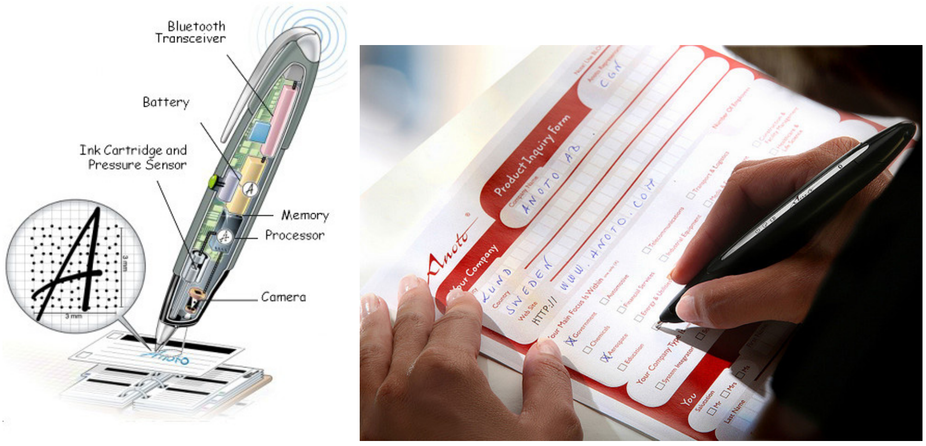


Figure 2.12: Anoto technology. Schema of the digital pen and the dot pattern (left) and a form document with embedded pattern to detect pen markings (right)

The scientific literature abounds with creative examples of how pen on paper interaction can be digitally enhanced, from simple digitising of ink content to advanced paper-based interfaces. Many of those achievements were made possible thanks to robust pen-tracking technology, at the forefront, that developed by the company Anoto [3]. Their technique relies on a combination of a unique dot pattern printed on a piece of paper (or other material) and a digital pen with an embedded camera that is able to determine its position based on the detected portion of the pattern beneath it (Figure 2.12). The tracking speed and accuracy are extremely high (between 70 and 100 Hz for the capture rate and 0.03mm for the precision [110]) which allows for high-fidelity digitising of pen data. Depending on the type of pen, input data can be transferred either continuously via streaming or in batches when synchronising with a docking station attached to a computer. Because the recognition of the dot configuration is performed by an IR sensor, as long as the dot pattern is printed with IR-absorbing ink and user input is performed with IR-transparent ink, the pen is able to distinguish between the two, which allows it to work on surfaces that already have printed content on them.

One of the first applications of digital pen and paper that met commercial success was a solution to streamline the processing of paperwork such as administrative forms and reports [87] (Figure 2.12 right). Other popular solutions include augmented books for education and interactive paper notebooks, where handwritten notes can be easily integrated into digital workflows (e.g. Adapx Capturx [2]). Some products such as Oxford's Easybook even allow users to specify commands directly on paper to link notes to particular tasks to be performed upon synchronisation (e.g. the generation of Word documents E-Mails, to-dos etc.) or, like Papershow, annotate and control presentations in real-time [18].

Most, if not all, of the commercial systems mentioned above were preceded by much groundwork in the research community. As early as 2003, for instance, Guimbretière introduced Paper Augmented Digital Documents (PADDs) [83], which showed how a cross-media loop between paper documents and their digital counterparts could be established so that editing could happen in either realm. Thus, annotations made on paper could be easily reflected on the digital content thanks to Anoto and conversely, modifications made on digital documents could be materialised on paper with new printouts. The PADD concept was extended two years later with PapierCraft [126], which added pen gestures to trigger commands, such as tagging, selection and copy-paste operations, creation of hyperlinks etc. Those ideas were exploited in paper-controlled presentation systems that perhaps inspired Papershow: PaperPoint [176] and PaperCP [125], as well as proofreading software such as Proofrite [61] and PaperProof [196] (the gestures developed for the latter to perform text editing and annotating operations are potentially useful for tabletop document-editing interfaces). There were also notable augmented paper applications deployed in the field. In this category, one can cite EdFest, an interactive paper-based festival guide with contextual audio information delivered to the user [36] and ButterflyNet, an augmented notebook system for field biologists that allow them to create links between various collected items and their notes [210].

For a large part, the development of the aforementioned research prototypes did not directly rely on standard vendor-provided SDKs, due to the latter's limitations and insufficiencies. Streaming, for example, was not readily available in Anoto's earlier pens, nor was the possibility to have multiple pens simultaneously connected to the same computer. Furthermore, developers felt that robust frameworks with higher levels of abstraction were needed to have more flexibility when building applications. Those reasons motivated the creation of custom toolkits such as PaperToolkit [211] and iPaper [175]. One should perhaps also mention the commercial SDK once provided by Livescribe [11], a manufacturer of digital pens, which allowed the execution of code directly on the pen via so-called penlets. This SDK, as of 2013, is no longer available.

Despite claims of bridging the analogue and the digital world, transitioning from one media to another is not entirely seamless and straightforward. Because paper lacks feedback capabilities, the proximity of a computer is required at some point to process and produce responses to the input, which does not make for a completely blended experience. Recognising those shortcomings, vendors such as Livescribe created digital pens with integrated speakers, a microphone and a small display, giving the possibility of immediate and in-place (albeit limited) feedback to the user when writing with the pen. Typical examples of in-pen applications are word translations, calculations and interactive quiz games (with results given either via audio or shown on the pen's display). Still, such a solution does not address the problem of visualising the full response of an action and possible content changes directly on the paper document. Researchers have attempted to tackle the issue with projector systems, either directly mounted on the pen (e.g. PenLight [177]) or on a separate unit (e.g. MouseLight [178] and FACT [129]). Those systems naturally cannot change the inked content on the pieces of paper, but they provide added-value through superimposed layers of new information (e.g. results of a full-text search, annotations made by other users etc.). Another influential piece of work worth mentioning, although it does not involve pen interaction, is PaperWindows [97]. This project considered sheets of paper as virtual screens on which windows of computer applications could be projected. Users could then execute various gestures on the pieces of paper to interact with the digital content.

Augmented paper interaction is an active area of research. A comprehensive survey of pen and paper systems can be found in [179].

2.4 Pen and Touch Interaction

As explained in the introduction, pen and touch interaction is a recent addition to the mix of NUI devices and has been the focus of a respectable amount of research effort in the last few years. It is important here to distinguish asymmetric from symmetric bimanual interactions, as in the latter case, the two hands contribute equally to the execution of a task, e.g. lifting a box, clapping one's hands etc. Pen and touch is inherently asymmetric, and even though technically the pen could be used as another finger, the fact that one hand is not interacting directly creates a dichotomy that changes the balance of the input context. Nevertheless, a pen and touch interface does not exclude parallel interaction patterns, as bimanual multitouch gestures are entirely possible with a pen-holding hand when the pen is stowed (examples will be provided further below). Pen and touch therefore also benefits from the manual and cognitive advantages of direct two-handed multitouch interaction [55, 119].

2.4.1 Asymmetric Bimanual Interaction

Guiard's Kinematic Chain Model

The study and application of pen and touch input have their roots in work done on bimanual coordination in the field of cognitive sciences [186]. Of particular influence is the seminal work by Guiard, who introduced the Kinematic Chain Model, a framework describing the asymmetric roles of the two hands [82]. Specifically, the model contends that the non-dominant hand (NDH) sets the frame of reference in which the dominant hand (DH) operates (right-to-left spatial reference in manual motion principle) and the DH works at a finer spatial and temporal level than the NDH (left-right contrast in the spatial-temporal scale of motion). In other words, the NDH performs coarse manipulations and sets the context of the precise actions executed by the DH. Guiard further observed that the NDH usually contributes earlier to the action than the DH (left-hand precedence in action principle), i.e. an initial setting of the frame of reference is required for the DH to start to work. The typical example illustrating how those principles apply is writing on a piece of paper. The NDH starts by moving the sheet and setting its position, then, the pen-holding DH performs the detailed task of writing (during which, incidentally, the NDH continuously holds down the sheet of paper). Occasionally, the NDH readjusts the position of the piece of paper.

This specialisation and complementarity of the two hands were confirmed in studies conducted by Kabbash et al. [103, 104] and Hinckley et al [90, 91]. The main lesson from those experiments for interface designers is that human motor skills need to be well understood in order to design effective bimanual interfaces with minimal cognitive load for the users. Two hands working in parallel does not necessarily translate into increased efficiency. As Hinckley et al. remarked, "there is a hierarchical structure to dexterous bimanual manipulation", which constrains how input and control mappings should be assigned to the two hands [90].

Asymmetric Bimanual Interaction in Practice

The principles of asymmetric two-handed interaction have been applied in a number of contexts, such as manipulation of 3D virtual objects [62, 90, 174, 214]. One project of particular interest, because it attempted to disrupt GUI conventions with novel bimanual input techniques in a real-world use-case where real users and stakeholders were involved, is T3, a tool for digital artists [116]. The design of the T3 system and its interaction model follows a philosophy aimed at maximising the screen space given to application data instead of cluttering the interface with a multitude of widgets as in traditional GUIs. The authors achieve this by implementing their chosen set of operations as bimanual gestures performed with two rotation-sensitive pucks (essentially two one-button mice) that can be moved on a tablet. The drawing tools are all contained in a single floating toolglass palette (based on Bier et al.'s Toolglass concept [41]) controlled by the NDH. The DH selects tools on the toolglass, following which the two hands are used together in a chunk [54] to create and set the appearance of new geometrical objects on the drawing canvas, e.g. a rectangle, whose size and rotation angle are determined by the two pucks. In a second step, the researchers integrate an adapted version of their prototype in the paint program StudioPaint to see how their interaction paradigm helps enhance artists' input vocabulary within an existing application. In [154, 155], Raisamo and Räihä describe a bimanual technique to position a virtual straightedge to perform alignment, carving and affine transformation operations on shapes. The ruler is controlled by a mouse (primary pointing device, DH) and a trackball (secondary pointing device, NDH), where the former is used to move the ruler and the latter to set its length and angle.

The problem with many of those early studies and systems is that their bimanual input techniques rely on proxy devices such as mice, pucks and trackballs, i.e. manipulations were performed indirectly and at a distance from the output display. In some cases even, those pointing devices had completely separate supports (two different tablets), creating a disconnection between the movement space of each hand. The consequence is that there was no common physical frame of reference, which can possibly mitigate the applicability of the generalisations of some of those results to other types of bimanual tasks. People have a strong kinaesthetic perception of their limbs and the possibility to move one's hands freely within a common input space possibly contributes to higher quality interaction. A multitouch tabletop provides a large shared space that fulfils that requirement, which is why it is particularly suitable for bimanual tasks, provided applications are designed to make use of those properties. Wu et al., for instance, show, within their gesture registration, relaxation and reuse model, that powerful gesture chunking and phrasing can be elegantly achieved on the tabletop to create fluid and efficient compound interactions [208].

2.4.2 The Pen and Touch Interaction Paradigm

As mentioned above, pen and touch interaction is a special case of asymmetric bimanual interaction. The DH handles the pen for indirect but fine-grained input, while the NDH interacts directly with the surface for coarser-grained manipulations. The two hands can work independently (which does not imply sequentially) or together for specific combined actions. But as has been established by the kinesiological studies, human motor skills are governed by certain rules, therefore the interaction vocabulary and design space for effective pen and touch interfaces are subject to constraints.

Properties of Pen and Touch Interaction

In [44], Brandl et al. initiate a reflection on the characteristics of pen and touch interaction and propose a rough taxonomy of input types for the two hands distinguishing aspects such as modes (inking or command), single point or posture (i.e. shape), static or gesture input. The authors also draw a table with the pros and cons of the two modalities. This analysis is pursued in more depth by Hinckley et al. [92, 93], who identify nine key attributes of pen and touch input, which are reproduced in Table 2-1. Generally, the researchers argue that a theoretical correspondence exists between the specialisation of the two hands in real-world tasks, such as writing on a piece of paper, and for digital transactions on a pen and (multi)touch device. Hence, perceptual mechanisms of humans in the physical world can be exploited to design more natural and effective interfaces. The combined availability of pen and touch in a single platform provides the foundations for this natural and rich design space. Particularly compelling is the possibility for pen and touch interactions to form phrases of multiple actions in compound tasks. Thus, combined interaction patterns, especially touch followed by pen gestures, yield rich transactions that integrate several substeps of unitary commands. Those steps are: (1) object selection, (2) pen-mode switching and (3) the possibility of immediately following with one or more pen actions to form a phrase. Examples of such compound actions are holding down an object + dragging it off with the pen to duplicate it, using shapes as dynamic templates or masks while drawing with the pen, putting a flat hand down to constrain pen tracing [44] etc. (more examples are given further below).

PROPERTY	PEN	TOUCH
Contacts	**1 point** *A single well-defined point.*	**1-10+ contact regions** *with shape information.*
Occlusion	**Small (pen tip)** *But hand still occludes screen.*	**Moderate** *("fat finger") to* **Large** *(pinch, palm, whole hand)*
Precision	**High -** *Tripod grip & lever arm for precision, writing, sketching.*	**Moderate**
Hand	**Preferred hand**	**Either hand / Both hands**
Elementary Inputs	**Tap, Drag, Draw Path, Crossing**	**Tap, Hold, Drag, Pinch, 2-finger Hold (thumb + index)**
Intermediary	**Mechanical Intermediary** *Takes time to unsheathe the pen. Pen can be forgotten.*	**None: Bare-Handed Input** *Nothing to unsheathe or lose. No lever arm. No buttons.*
Acquisition Time	**High** *(first use: unsheathe pen)* **Moderate** *on subsequent uses: tuck pen between fingers.*	**Low** *No mechanical intermediary to acquire.*
Activation Force	**Non-Zero** *Tip switch/ minimum pressure.*	**Zero.** *Contact area (a proxy for pressure) often can be sensed.*
False Inputs	**Palm Rejection** (while writing) *Palm triggers accidental inputs, fingers drag on screen, etc.*	**Midas Touch Problem** *Fingers brush screen, finger rests on screen while holding it.*

Table 2-1: Hinckley et al.'s tableau of design properties for pen and touch [93]

Informed by a study in which they observe how people work with paper notebooks, clippings and pens, Hinckley et al. elicit a number of design principles that can be summarised as: the pen writes, touch manipulates and pen+touch provides new tools. In other words, in unimodal situations, the pen inks (marks the surface as would happen with a physical medium), touch carries out common operations (panning, zooming, selecting etc.) and in combined mode, pen+touch performs other functions. Those pen+touch operations, however, are constructed from unimodal primitives. For the pen, the authors use tap, drag-off, crossing or drawing a stroke and for touch: single-finger tap, single-finger-hold, holding with two fingers and crossing. The authors do not map all combinations to particular gestures. In their proof-of-concept prototype they mainly use the NDH to hold items in reference to which the pen then acts. This is in accordance with Guiard's right-to-left spatial reference and left-hand precedence principles.

Pen Mode Switching

The fact that a digital pen may be used not solely to mark a medium as in the physical world is evident. In unimodal interfaces, the pen serves as replacement for the mouse and therefore executes all kinds of non-inking operations. Even in pen and touch environments, the pen's role is not restricted to that of a monofunctional drawing or writing utensil, as has been established previously. This poses the question of how the function state of the pen or mode can be changed effectively without disrupting the task to be executed. A study by Li et al. comparing mode switches triggered by the pen-holding DH and the NDH revealed that transitions activated by the latter yielded higher performances (and presumably also a lower cognitive load for users) [123]. Earlier work by Sellen et al. showed that user-maintained modes to control an input device's state are more effective than system-maintained mode states [173]. The muscular tension required to keep a non-regular mode activated works as a kinaesthetic awareness mechanism and hence minimises mode errors. Those findings are consistent with Guiard's kinematic chain model.

Mode-switching gesture designs adopted so far have shown varying degrees of adherence to the strict division of labour and role assignment of modalities advocated by Hinckley et al. Some researchers seem to have based their design decisions on their interpretation of task and operational requirements and context analyses rather than rigorously following those principles (which is not necessarily a bad thing, since there is no consensus on best practices for pen and touch interaction). Wu et al. for instance, choose the pen's default (i.e. non-modified) mode to be for executing object manipulations and writing mode has to be specifically activated with a two-finger posture [208]. Zeleznik et al. use touch-driven pen gestures, whereby to execute a command the pen starts the gesture and activates a feedforward widget to be confirmed by touch, if a pen gesture stem (that is, a subset of a complete gesture) is recognised [213]. This technique violates Guiard's KC model, as the DH here not only sets the frame of reference but also initiates the action. In most cases, however, pen-mode switching is performed according to the KC rules, i.e. the NDH first sets the mode and then the pen acts within the selected context. The mode-setting action can occur in different ways: depending on whether a particular object will be directly affected by the operation, fingers of the NDH will be placed on that object (for instance, to copy it) or not. For pen and touch manipulations not immediately targeting specific objects, the NDH typically activates mode changes with special postures performed on free areas of the interface [44, 77] or with available function widgets such as buttons or gesture panels [85].

2.4.3 Pen and Touch-Enabled Apparatuses

As we have seen, pen-sensing digitisers and touch-sensitive devices have been available for several decades. A 1987 IBM patent describes a "combined finger touch and stylus detection system" in which touch is detected capacitively and the pen by electromagnetic radiation [81].

It does not seem like this patent was turned into a product or a prototype that was publicly demonstrated. Nonetheless, for several years thereafter, stylus and touch input types were either considered individually or interchangeably so that the two modalities were never distinctively present at the same time. Systems like the Active Desk [52] and Dual Touch [137] introduced some level of pen and touch interaction but with several limitations. In 2004, Yee created a compact, self-contained prototype integrating differentiated pen and touch-sensing capabilities [209]. His implementation consisted of a regular stylus-based tablet PC augmented with an external resistive touch-sensing panel. Since the touchscreen sensors also captured pen input, a special software driver was required to determine the coordinates of the touch point (the touchscreen did not support multitouch and reported a contact point interpolated from all detected inputs).

In 2007, a large rear-projected screen supporting simultaneous pen and touch input was achieved with INTOI [45]. The system combined a special foil with an Anoto pattern overlaid on the screen for pen-sensing and an IR-tracking setup to recognise hand gestures (but not fingertips). Anoto was again used by Brandl et al. to create a pen and multitouch tabletop, where this time the foil with the dot pattern was laid over a DiamondTouch screen, thereby enabling finger-driven multitouch gestures [44] (Figure 2.13 left). Brandl et al. subsequently created a rear-projected and tilting version of their system (with an FTIR-based display) to support different orientations with a single interactive surface [121]. Being relatively straightforward to construct, this type of solution coupling multitouch tabletop hardware with Anoto technology was adopted by other researchers as well [75, 130]. Other than boasting precise, high-resolution pen-digitising and responsive multitouch detection, the DiamondTouch + Anoto combination has the further benefit that it provides support for multi-user configurations almost "out of the box", as both the DiamondTouch and Anoto pens have built-in mechanisms to uniquely and reliably identify multiple users. So far those types of scenarios have not been considered and so it remains to be seen if and how this capability will be exploited.

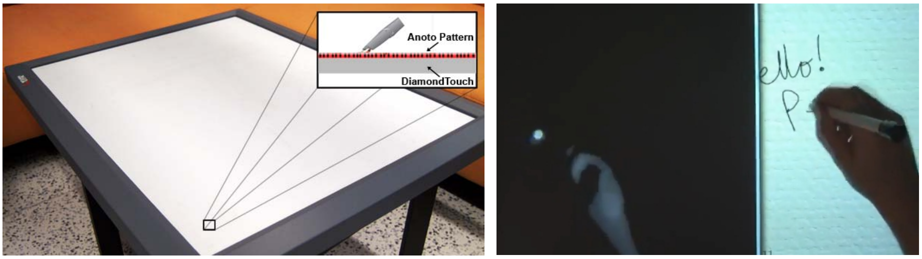


Figure 2.13: A DiamondTouch augmented with a foil with an Anoto pattern [44] (left) and pen distinction on an IR surface by brightness [93] (right)

For touch surfaces based on infrared sensing (such as Microsoft PixelSense), stylus interaction can be added with digital pens that, when actuated upon pressure, emit IR light [92, 213]. Since that light is much brighter than finger contacts, it is easily identified in the raw image frames returned by the sensors (Figure 2.13 right), although misdetections can happen [213]. A more sophisticated IR pen supporting different pressure levels as well as user authentication was engineered by Qin et al. [153]. More recently, Hamilton et al. built an apparatus based on a Wacom Cintiq tablet and their ZeroTouch multitouch sensor [85, 144]. This was just before Wacom introduced their Cintiq 24HD model with integrated pen and touch support (Figure 1.2 right).

2.4.4 Pen and Touch Applications

Several demonstrators and applications have been developed to showcase the advantages of pen and touch interaction. Following is a short review of the most prominent examples, with a particular focus on functionality achieved through the combined use of the two modalities.

Subsequent to the elaboration of design principles for pen and touch input [44], Brandl et al. created a painting program, which includes a number of hybrid gestures. A particular aspect that is emphasised in their prototype is the supplemental nature of the pen that allows finer control of operations started by the NDH. For instance, touch summons a menu and the pen makes precise selections such as picking a colour or performing a specific number of undo operations. The pen can also be used to carefully position cut-out elements coarsely moved around by the NDH. Similar to T3 and its shape creation tool through chunked actions, one of the rectangle selection techniques involves phrasing the menu-calling command and the actual selection operation: the first corner of the selection rectangle is set by the menu-triggering finger and the second corner is determined by the pen. For polygonal selections, the pen is used to trace successive segments, whose vertices are set by taps of the NDH.

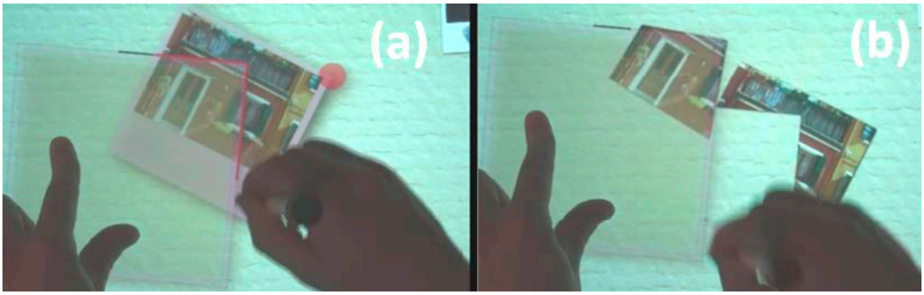


Figure 2.14: Using an item held by the NDH as a straightedge to cut out items with the pen in Manual Deskterity [92, 93]

In an analogous approach, Hinckley et al. apply their general design considerations for pen and touch interaction in a prototype called Manual Deskterity, a scrapbook application [92, 93]. As with a real scrapbook, pictures, clippings and other pieces of documents are gathered, cut out and assembled to form an album in a fluid manner. The authors underline the continuity of the manipulations and the effortless interleaving of different touch and pen actions in a smooth flow, e.g. to drag out a sticky note from a bezel menu or turn a page with the NDH and immediately start writing on it with the DH. The new tools they introduce, i.e. those requiring the combined use of the two modalities, include a touch hold + pen drag off gesture to create duplicates of elements, a hold + tap action to pile selected objects, using the pen as a utility knife to cut out image pieces held down by the NDH (other images can also be used as straightedges to cut along their contours when two fingers are held down (Figure 2.14)) or as a brushing tool to trace copies of selected image portions and turning parts of freeform pen strokes into tape curves on-the-fly by tapping them with a finger. The researchers also describe a technique to pull out in-place menus such as a colour palette from finger shadows, which are virtual traces momentarily left by finger taps. Interestingly, following observations in user tests that most people expected to be able to smear off colours from the palette with their fingers, they added this feature, thereby knowingly breaking their pen writes + touch manipulates design rule. This exception echoes another concession the authors make following their test study: for common controls and widgets (menus, buttons etc.), the possibility to use pen or touch interchangeably should be given to the user.

In [75], Frisch et al. present a diagram-editing application that includes a set of pen and touch commands resulting from an analysis of user-elicited gestures. Except for a few hybrid interactions such as finger hold + pen drag to create copies and a finger tap + pen select to connect nodes, all their gestures are unimodal. Moreover, the stylus is not only used to draw nodes and edges, but also for lasso selections and delete gestures, this without explicit action to switch to a command mode. With the limited gesture set of their diagram editor, the choice not to clearly disambiguate inking and command modes has little impact on possible misrecognitions and errors, but for applications relying more extensively on pen-based interactions, such a design may lead to an increase in conflicts between the two types of modes. In a subsequent publication, Frisch et al. introduce NEAT, a set of more advanced tools and gestures to execute a variety of layout and alignment operations [77]. Those tools include interactive on-screen guides invoked by hybrid touch and pen gestures or by tangibles such as set squares. In the first case, a finger touch creates a disk on the surface, from which curves can be traced with the pen. Those curves then form the guide along which objects can be positioned. In the second case, the tangible's shape directly determines the guide's geometry. Objects to be aligned can then gently be pushed towards the guide to "snap" to it. Multitouch gestures are also available, e.g. holding an element with two-fingers and tapping other components within a group automatically aligns and evenly distributes them along the axis formed by the two end objects.

An application domain that, like diagram-editing, lends itself particularly well to pen computing is math and equation writing. Zeleznik et al.'s Hands-On Math blends natural pen interaction with the power of a computer algebra system in a cohesive pen and touch interface [213]. The system is supported by a set of ad hoc gestural techniques to recognise handwritten mathematical expressions and perform various algebraic transformations. Hands-On Math also includes some document-editing functionality to create annotations, issue search queries and reuse content from web sites. Here as well, most of the interactions are sequentially unimodal and some gestures are triggered without explicit switching to a command mode, but the smooth interweaving of inking actions to input mathematical expressions with touch manipulations to edit them makes for an integrated and efficient interactive environment for the purpose of the task. Their touch-activated pen gesture technique is used for commands such as rectangle selection, space insertion and clipboard pasting. As mentioned above, with this method, the pen initiates the command (e.g. a crop mark for rectangle selection) and touch parameterises (drags the second corner of the rectangle) or confirms the action.

Taking up another popular pen-based activity related to note-taking and scrapbooks, namely active reading, Hinckley et al. develop GatherReader, an e-reader application with support for annotating and content gathering [88]. The main design concern here is to integrate instruments to smoothly and seamlessly perform the secondary task of extracting and organising content within the primary task of reading without disrupting the latter. The authors achieve this with a combination of multitouch and pen actions, with a mostly strict adherence to the division of labour principle. Touch is used to flip pages, select page chunks, drag content and operate widgets, while the pen is used almost exclusively for inking. The only exception is to highlight phrases, after a passage has been selected using a two-finger scope-framing interaction (which functions as the mode-switcher). There are no simultaneous pen and touch gestures, as the focus is more on interleaving unimodal actions rather than combining the two input types.

In the InfoVis community, researchers have also recently explored the potential of the pen and touch interaction paradigm for the presentation and analysis of chart data on vertical screens such as whiteboards. SketchInsight is a tool with which users can manipulate bar graphs, line graphs and scatterplots mostly using sketch-based pen gestures (e.g. L-shapes to draw graph axes, a circle for a pie chart etc.) [193]. In this prototype, all interactions are unimodal, as per the authors' design choice to minimise cognitive load imposed on users. Thus, touch is used solely for navigation and item selection and there is no mode-switching.

Lastly, Hamilton et al. apply pen and touch interaction to real-time strategy gaming [85]. Since this type of context does not involve inking (or simulated paper-based desk work for that matter), the pen is exclusively used for commands, the types of which are determined by NDH selections and multi-finger postures (or "chords", to borrow Westerman's terminology [199]) executed in dedicated edge panels of the interface. Within that area, a command selection panel is available to initiate specific functions, according to the number of fingers placed on the pad. The pen then articulates the selected command either with single-point selections or with strokes (a technique the authors call "bimanual overloading"). Multiple commands can be queued by sliding the chorded posture to the right so that the rightmost finger enters the nearby queuing activation panel ("scrubbing") (Figure 2.15 left). Less common functions have their own buttons that can be triggered with simple taps. Queuing, for instance, is carried out by holding the button of a function and using the pen to issue successive commands of the same type. This interaction method is similar to Hinckley et al's springboard technique [89]. The authors point out a further feature of their application: pen-in-hand touch actions, i.e. touch gestures performed by the DH with the stylus stowed away. Those manipulations are mainly used to adjust the camera position with classic multitouch interactions to pan, zoom and rotate the view. A three-finger pinch gesture is also available to control pitch.

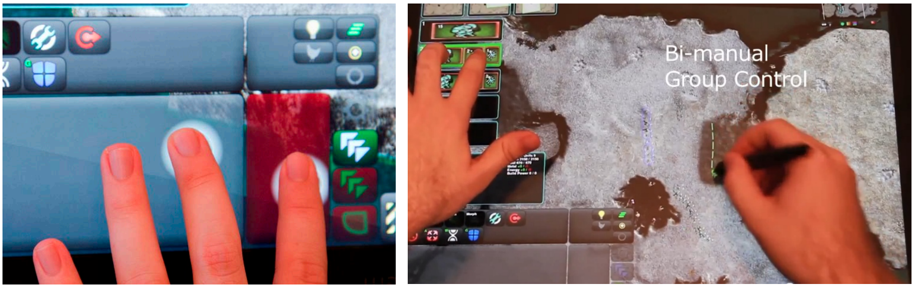


Figure 2.15: Hamilton et al's pen and touch real-time strategy gaming environment [85]: queuing commands by sliding a multi-finger chord [199] to a hotspot area (left) and issuing commands with the pen to a group of units associated with specific touch buttons (right)

2.5 Analysis

Digital workdesks, as part of the equipment of the office the future, have evolved from hefty pieces of furniture with slow and limited interactive capabilities to sleeker, more powerful and responsive workspaces. Those include pen and touch-operated digital tabletops, which have become more accessible, either as experimental systems put together by researchers or very recently as commercial products. This maturity of the hardware has enabled practitioners to concentrate more on the design of novel interaction techniques and applications. In the particular case of pen and touch, the design of interfaces and gestures has taken place on the background of many studies on human motor skills. Early on, Guiard's kinematic chain model established a relatively sound and robust theoretical framework characterising bimanual interactions. Yet the literature shows that researchers do not always strictly adhere to it, as, in some cases, they choose to implement ad hoc interaction models based on their own design considerations and the application context. Thus, the NDH does not always start the action, nor does it invariably set the frame of reference. This discrepancy is perhaps due to the fact that empirical rules governing interactions observed in the physical world do not necessarily all translate to digital and virtual contexts. A piece of paper does not react to touch input and regular pens have only one mode: inking (and sometimes erasing). Computer interfaces on the other hand are able to provide a wide range of feedback to user input. Users seem to perceive the difference as many of them, for instance, consider the pen to be as valid a selection tool as the finger [75]. In those contexts therefore, it appears wiser to construe the principles of the KC model as guidelines rather than rigid design rules. One axiom that was consistently verified in the reviewed work, however, was the stylus being used for high-precision tasks, such as handwriting, drawing, lasso selections, path-tracing etc. This choice is obvious considering the differences in surface contact size between a pen nib and fingers.

Concerning pen-mode switching, systems do not adopt the same mechanics as well. Mode changes can either be triggered implicitly via detected pen gestures [75, 213] or explicitly through NDH postures [44, 85, 92]. If the likelihood that inked content will conflict with a gesture is minimal, the former alternative is appealing, as it is more direct and requires no bimanual coordination. But if the gesture set is more extensive or if more robustness is desired, NDH-based mode control is a compelling solution, especially in the context of chunking and phrasing bimanual actions [54]. A common ambition of those projects, however, is to push the envelope of the interactional space, but without overstepping the bounds of acceptable gestural complexity for reasonably dexterous users. Hence, an application can feature a number of bimanual commands, but those must not be overly difficult to execute. A tacit rule that can be educed from the state of the art is that gestures should not require both hands to be in motion simultaneously. Again, this makes perfect sense, considering synchronised movements of the two hands would be much like patting one's head and rubbing one's stomach at the same time. Moreover, the unitary interactions that form two-handed compound commands are extremely simple. Hinckley et al. use only finger hold and pen drag actions as the basic elements of their "new tools". This does not mean that slightly more sophisticated gestures should not be attempted. Hamilton et al.'s chord-based command mode selection and scrubbing techniques, for instance, show how the mode-setting NDH need not necessarily remain still to perform rapid and effective context-switching. Interesting also is Brandl et al.'s dual-precision element-positioning method making use of modalities as coarseness differentiators.

In terms of document-related tasks and interactions, prototypes developed so far (including those for augmented paper) illustrate how elementary editing and annotating operations can be carried out. As exemplified by Manual Deskterity, Smart Publisher etc., typical document authoring functionality in pen-based applications revolves around gathering and reusing external content, although it is not always very clear where the material is coming from and how it is retrieved and sorted out. Data directly input by the user is usually limited to handwritten notes and simple drawings. More advanced functionality is rarely included, as the development overhead would be too great and the rewards in the research community for feature-rich systems are arguably not worth the effort. Still, there is something to be said for holistic approaches that aim to create usable tools meeting real-world requirements formulated by real users. T3 and Hamilton et al.'s real-time strategy game are prominent examples of that. Stakeholders engaged with a novel interface and interaction methods provide informed feedback as to how the proposed system is able to help them accomplish their work more efficiently or more easily. In the case of document engineering and document creation, a pen and touch editor meaningfully improving on prior art should therefore include a more extensive toolset with which users can produce reasonably elaborate documents.

3 Pen & Touch Software Framework

Architecturing multimodal applications presents a number of software engineering challenges, all the more so if modalities can be simultaneously active on the same interaction space, as in the case of pen and (multi)touch input. In particular, the management of complex hybrid interactions and gestures needs to be robustly supported by appropriate software constructs. Due to the possible interplay within and between modes, it is not sufficient to simply glue input processors for each interaction type together, although this may be a starting point. A tempting approach would be to build on an existing multitouch framework [69, 118, 157] and integrate support for an additional pointing device into its processing pipeline, but this requires significant reengineering and may not necessarily be worth the effort, especially if additional sophisticated interaction patterns also need to be implemented.

Most frameworks include default functionality and behaviour that cause components to react to (unimodal) gestures and interactions in relatively standard ways, which may not be suitable for the kinds of pen and touch requirements considered here. For instance, current frameworks typically associate a single object on which a particular gesture is performed, usually the object touched on the first contact. Already in a multitouch environment this might prove quite restricting, when a gesture needs to be defined on multiple objects that are touched simultaneously or successively by different fingers or hands. With bimanual pen and touch, this case can occur even more frequently (since there is an incentive to use combined interactions) so that it becomes essential to allow for more flexibility when considering UI objects associated with gestures. Furthermore, the majority of available multitouch frameworks mostly provide support for online gestures only, i.e. interactions upon which the system reacts immediately, such as panning, zooming, rotating etc. [107]. Handling of offline gestures, which trigger responses only after they are completed (or detected), is often absent or limited to single-stroke gestures, let alone hybrid pen and touch gestures. A further constraint of such frameworks is that touch contact shapes are generally limited to ellipses of two types: cursor (finger) and blobs, and tagged objects (fiducials). This characterisation of input types is also the one used in most implementations of the popular TUIO protocol [106] (version 2 of the protocol introduces the possibility to define the geometry of contact shapes much more precisely [105], but those features are currently not supported by major frameworks). This precludes gesture signatures based on arbitrary contact shapes such as flat hands, chops, clenched fists etc. [74, 200] that can be used for pen-mode setting. Of course, such gestures can still be created if access to sensor data is provided, but this requires custom processing of raw information, which defeats the purpose of relying on a framework in the first place.

Owing to the particular requirements of pen and touch in the considered context and the limitations of current frameworks, it makes sense to invest some effort to build a bespoke software foundation with a set of dedicated tools to facilitate prototype development. This chapter describes the custom framework created for that purpose.

3.1 Requirements

3.1.1 Goals and Scope

Although the primary goal of this endeavour is to design a structured, reusable codebase for prototypes developed within the context of this project, i.e. for applications running on pen and touch tabletop surfaces, the architectural concepts and techniques tackled here are general enough to be applicable in a wide range of scenarios involving multimodal surface interaction, which is why it is deemed as a contribution in and of itself.

In terms of target platform, the software is intended to be used for applications running on the system adopted for prototype deployment and testing, that is, a DiamondTouch screen combined with Anoto technology. That is not to say that the framework should be tightly coupled with the hardware, as that would immediately limit its scope. As with other libraries that aim to be portable and platform-independent, the inclusion of multiple layers of abstraction allows components to be modified and changed with minimal adaptation effort. Considering the specificities of tabletop systems and multitouch technology (in particular the DiamondTouch's limitations in terms of single-user multitouch detection [38]), the framework should make sure the handling of sensor data remains relatively isolated from the upper layers of the architecture.

With regard to gesture-supported UI components, those should include standard behaviour controlled by classic multitouch gestures (dragging for translation, pinch/spread for zooming-rotating) but with the possibility to deactivate, overwrite and overload individual actions, depending on the needs of the application. Similarly, if the component can be marked by the pen, controls should be integrated to manage inking, according to configurable rules that may include different styling options for strokes and possibilities to process said strokes before display, for instance, to beautify sketched content and convert it to typeset text or smooth vectorised shapes. Furthermore, components should be able to react differently if particular input combinations or gestures (unimodal or multimodal) are detected, as when the pen mode is changed by an NDH posture and pen data should not be interpreted as inking strokes but as commands.

Naturally, to serve its purpose in several instances, the framework should provide flexible means to adapt, extend and add functionality. As with any self-respecting GUI toolkit, components should be easy to create and offer customisation possibilities either via direct variable setting, method calls or through inheritance. Similarly, gestures should consist of standard, ready-made objects but also allow for user-specified adaptations to be effortlessly integrated. In particular, appropriate recognition engines need to be available, in order to facilitate detection of a wide variety of gestural patterns. Because interactions can potentially be bimodal, both pen and touch data need to be supplied to recognisers and gesture templates. Convenience methods to check whether a determined touch posture (such as a particular chord [199]) is currently active can also be included as quick shortcuts for gestures that can only be triggered in specific modes.

3.1.2 Formulation of the Requirements

Given the objectives set forth above, the following requirements can be formulated:

- Abstraction of hardware-dependent input data and clear separation of the modes, even if processed by the same digitisers (the latter case occurs, for instance, for vision-based systems that distinguish pen input from touch in a raw image through segmentation [92, 213]. This step, if necessary, should ideally happen at the driver-level).
- Input managers return rich objects encapsulating detailed data extracted from sensors. For touch, this means that continuous contact shapes should not be mapped to ovals as in the TUIO protocol (v1.1), but should be high-fidelity models of that information (for more elaborate interactions including the sensing of above-surface gestures, a 3D object model could also be considered). In the case of the DiamondTouch, this is more complex, because horizontal and vertical 1D signal arrays are returned by the edge antennas instead of 2D contact shapes. While a bounding box encompassing the whole contact area can be built from that data, a rectangle does not model sufficiently fine-grained information to allow detection of multiple fingers, let alone arbitrary shapes. Therefore the touch input model for a single frame here is two sets of segments representing the projections of touch intensities on the two axes of the system (see Figure 3.2).
- A component hierarchy with increasing levels of function-integration and sophistication. Components or abstract skeletons thereof at the lowest level simply receive pen and touch events passed on by a component manager which has access to input data, whereas higher-level elements include default translation, scaling and rotation behaviours driven by standard two-finger interactions or control handles as used in graphics software.
- A UI manager, which controls the layout and appearance of registered UI components and widgets. In particular, it should be able to handle elements that are "always-on-top" such as overlays and gesture strokes.
- A centralised and hence decoupled content transfer mechanism, which allows user elements to be copied or moved between registered UI components.
- An integrated gesture package containing pre-defined gestures but also flexible tools to create new ones. Those tools should include recognisers that continuously track pen and touch input data and with which specific stroke or movement patterns can be detected.

Both online and offline gestures should be supported so that components can react during, as well as after completion of particular user interactions.

– To guarantee a certain level of flexibility, the framework should include several loosely coupled layers representing the separation of concerns. Hence, when necessary, inter-layer communication should take place as much as possible between entities of adjacent layers.

3.2 Architecture

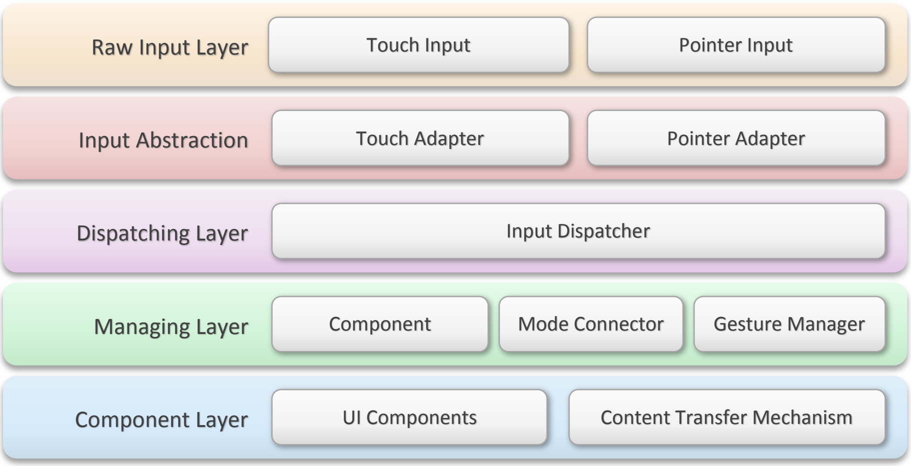


Figure 3.1: Architecture overview

Figure 3.1 shows the general architecture of the proposed framework with its constituting layers, which are described successively in the following subsections. Not represented in the figures and listings are the various utilities to perform redundant operations and computations (e.g. geometrical calculations to determine shape and line intersections for hit-testing, projections, stroke simplifications etc.) as well as elements responsible for managing resources and settings.

The following subsections describe the layers one after the other. UML diagrams are provided to show the different relevant sub-architectures as well as Java code examples to illustrate possible implementations.

3.2.1 Raw Input

At the lowest level, raw input data is collected through system or vendor-provided APIs, which expose digitised values of analogue signals produced by device sensors (with possible intermediate processing steps to transform and adapt the data to user space). For touch input, the data is typically delivered as frames streamed at regular intervals when a surface contact is registered by the system. The frequency at which those frames are updated depends on the hardware. In the case of the DiamondTouch, this frequency is 75 Hz [65], i.e. the application receives a theoretical maximum of 75 input frames per second. In practice, the real frame rate is lower, as input signals have to go through a long and complex processing pipeline, which can be affected by several external events.

APIs of multitouch devices commonly provide input data through series of events capturing the main phases of a touch action. Typically, those are TouchDown (when a contact is first registered), TouchDuring (fired continuously during the contact) and TouchReleased (when the contact is not detected any more). If the hardware is able to distinguish touch points, those events are fired for each individual contact area, i.e. multiple fingers touching the surface each trigger their respective TouchDown, TouchDuring and TouchReleased events. Correspondence of touch points between different frames is handled by the vendor API, which also assign them unique ids or indices to be able to track them easily. In addition to touch point coordinates, some toolkits also return information capturing finger orientation, which can be easily inferred from the contact shape geometry [194].

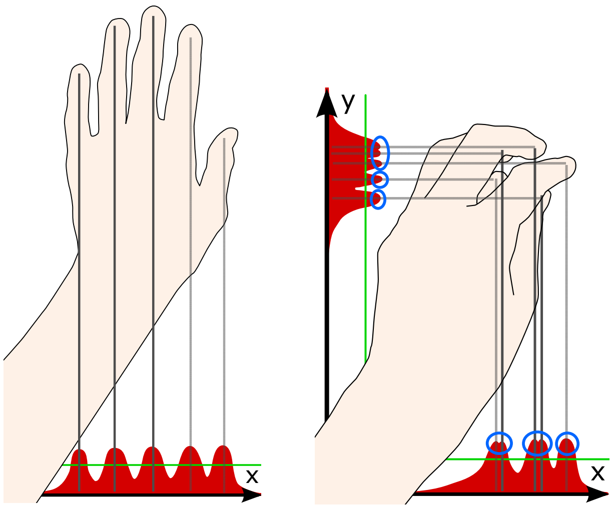


Figure 3.2: Detection of a 5-finger chord on the DiamondTouch.
Left: fingers are sufficiently spread out and parallel to the x-axis. The table's horizontal antenna is able to distinguish the 5 fingers.
Right: fingers are not aligned and the projection paths of some contact points overlap, resulting in merged peaks on both axes. Antennas cannot detect the presence of 5 fingers.

Because the DiamondTouch is not capable of reliably differentiating distinct contact shapes, its event model is limited to returning global touch status. That means: TouchDown is fired upon detection of a first touch on the screen and TouchReleased when no more active contact is registered by the tabletop sensors. In between, TouchDuring events are continuously emitted, regardless of the changes in the location and distribution of contact shapes. Developers who still wish to track different touch points therefore have to resort to custom techniques, for instance, using event history and heuristics [38]. But those solutions are far from perfect, as they are unable to cope with situations in which touch points occur simultaneously. A compromise can be achieved by relying directly on the two antenna signals. In most cases where users have multiple fingers placed on the surface, axis projections can still reveal their number and rough position. This information is generally sufficient to define mode-setting postures based on such touch patterns. A condition for chords to be reliably identified, however, is that the projections of contact areas yield the same number of above-threshold peaks on one of the axes. This can be simply achieved by laying fingers parallel to one of the table edges and spreading them sufficiently far apart (Figure 3.2 left).

For pen input, the event model is very similar to that of the mouse, including button pressed events if the stylus has a button on its side. Here as well, depending on the pen's capabilities, input data may contain more than just pointer coordinates, e.g pressure, tilt and azimuth, information that can be used to create close-to-real digital ink and interactions (e.g. for art applications) or to control operation parameters [40, 185, 189]. Anoto pens currently only include pressure as extra input data. In most situations, however, pen coordinates are sufficient for application needs. Besides, in a pen and touch context where both hands are used together and hence users' concentration is already in relatively high demand, one must be careful not to add too many degrees of freedom to interface controls in order not to cause excessive cognitive load.

Finally, both pen and touch –or rather the bare hand– can also be used for contactless input, i.e. above-surface interaction, or hovering. This adds further dimensions to the interaction space that can be exploited for gestures, as exemplified in several research projects [28, 135, 182]. While potentially an interesting input space, it is not yet clear how mid-air interactions can be successfully applied to tabletop interfaces in practical real-world cases. The framework therefore focuses on surface-bound pen and touch interaction, i.e. 2×2D (or 2D + 2×1D for the DiamondTouch) with immediate extension possibilities to include stylus pressure and tilt as further input parameters. In the future, when hardware with robust and reliable support for 3D interactions become available, integration on the software level can be added by implementing relevant input processors and including associated events, for example on the model of the Kinect [9] or Leap Motion [10] SDKs.

3.2.2 Input Abstraction

This step deals with the construction of abstractions or models of raw input data in order to encapsulate that information in hardware-independent, easily accessible and manageable objects. The construction of those abstracted input objects is the role of Touch and Pointer Adapters. As shown in Figure 3.3, both touch and pointer data objects realise a `UserData` interface, which associates input data with user IDs. As mentioned in the Background Chapter, both the DiamondTouch and Anoto can distinguish between different users and their SDKs, where distinct IDs are assigned to different users and pens respectively. Other tabletop devices without explicit hardware support for distinguishing touch or pointer interactions issued by different users have to rely on other means for identification (e.g. using hand features [156, 169] or even the users' shoes [163]). Although the prototypes developed for this project are designed for single-user situations, support for user-separate input was integrated, in case the framework is used for collaborative application development in the future.

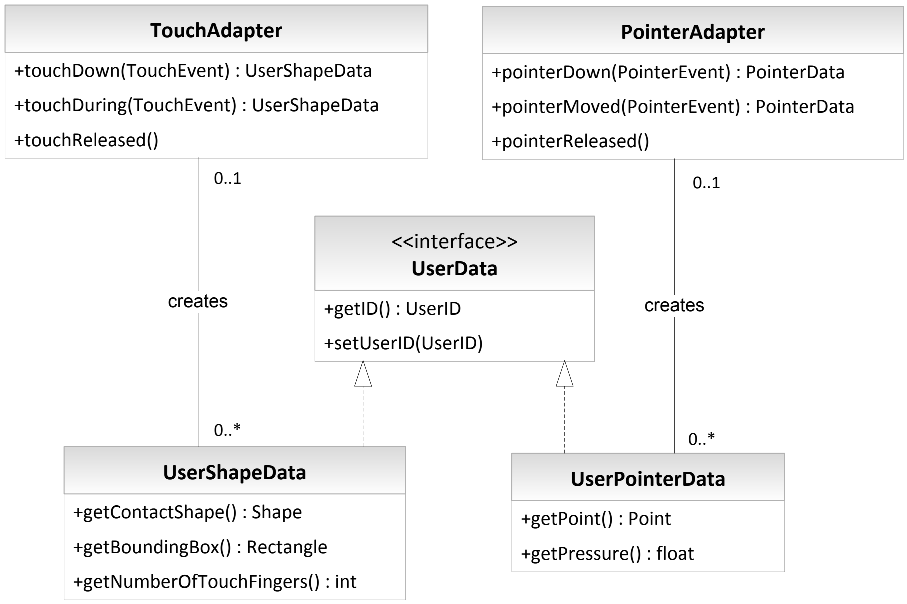

TouchAdapter
+touchDown(TouchEvent) : UserShapeData
+touchDuring(TouchEvent) : UserShapeData
+touchReleased()
PointerAdapter
+pointerDown(PointerEvent) : PointerData
+pointerMoved(PointerEvent) : PointerData
+pointerReleased()
0..1
0..1
<<interface>>
UserData
+getID() : UserID
+setUserID(UserID)
creates
creates
0..*
0..*
UserShapeData
+getContactShape() : Shape
+getBoundingBox() : Rectangle
+getNumberOfTouchFingers() : int
UserPointerData
+getPoint() : Point
+getPressure() : float

Figure 3.3: The Touch and Pointer Adapters with their respective Input Data Objects

Touch Adapter

Defining a common object structure containing relevant touch information is not always straightforward, due to the heterogeneity of devices and disparities in their sensing capabilities (especially for tabletops). As we have seen with devices like the DiamondTouch, descriptors of multitouch input frames cannot use individual contact points. A common denominator that can be derived from most, if not all major tabletop systems, including the DiamondTouch, is the smallest bounding box containing all touch points. A great amount of interactions can already be implemented with this object, including common actions for panning, scaling and to some extent rotation. Many shape-based postures and gestures can also be derived from the geometry of the rectangles. For interactions relying on specific chords, i.e. a certain number of fingers, on the DiamondTouch, it is necessary to have recourse to the arrays of axis segments.

Considering the requirements outlined above, a `TouchAdapter` class is defined to function as the bridge (meant as the software design pattern [78]) between vendor-specific sensor data and abstracted input. The Adapter contains the standard `touchDown`, `touchDuring` and `touchReleased` methods responding to global touch status. The first two functions process the hardware-dependent data out of which a neutral `UserShapeData` object is constructed. The class contains three methods, the most dependable of which is `getBoundingBox` as it returns the bounding box defined above. Two further methods are included to support more complex interactions such as finger and shape-based NDH postures: `getNumberOfTouchFingers` and `getContactShape`. Those functions represent a compromise between client requirements and specificities of the tabletop hardware to support. Implementations of the `getNumberOfTouchFingers` method therefore either use 2D raw data to reliably determine fingertips through segmentation, when 2D frames are available (as on the PixelSense, for example), or, in the case of the DiamondTouch, infer the most likely number of fingers via the antenna segment arrays. Similarly, for the contact shape, this is either a set of polygons of connected components encapsulating the different contact areas as obtained by analysis of the raw image, or simply the global bounding box returned by `getBoundingBox`.

Thanks to this design, developers can implement interactions based on a common set of input information (the bounding box and the number of fingers), while also retaining access to more precisely defined contact shapes if need be.

Pointer Adapter

The Adapter for pointer data is more straightforward as there are no ambiguities to resolve for input coordinates. Input data is captured in a single point, with possible inclusion of other information such as pressure, orientation and azimuth.

Implementations of `PointerAdapter` exist for the Anoto pen, which includes pressure data, and the mouse.

3.2.3 Input Dispatching

The Input Dispatcher is responsible for the initialisation of selected input adapters and the routing of input events to the various managers of the upper layer. It is, in essence, the glue between the lower input layers and the higher-level components of the framework.

One of the most important roles of the Dispatcher is to control the flow and order of how input events are passed on to the managing entities. In the current implementation of the framework, those entities are the component manager, which governs UI components, the gesture manager, which control standard and user-defined gestures and the Mode Connector, which encapsulates the binding of specific pen modes with particular touch input signatures. (those objects are described in detail in the following subsections). The dispatching order for input event (both pen and touch) is: Mode Connector → Gesture Manager → Component Manager. The mode connector receives input data first to determine the currently active mode based on the detected posture. Then, the Gesture Manager takes the input and passes it on to all the registered gestures. Online gestures, there, have a chance to consume the event, that is, to stop the propagation of the input event to other elements, should this be required by the application. Such a case can occur, for example, if a particular type of gesture is in the process of being executed and other components should not react to the input. Finally, input data is transmitted to the Component Manager (unless the event has been consumed in the previous step), which routes it through its internal component chain.

Since managing layers also require access to fine-grained input data (to process posture and gesture, among other things) and client objects should not directly read that information off the input adapters, the Input Dispatcher also includes static methods exposing the number of contact fingers and the contact shape in the last input frame. Typically, however, those methods are intended to be used in the input processing chain of components, i.e. within `touchDown`/`pointerDown`, `touchDuring`/`pointerMoved` and `touchReleased`/`pointerReleased` functions.

3.2.4 Managing Layer

This layer contains the various managing entities, which organise and process the input received from the dispatcher. For the sake of clarity and conciseness, the functions handling the routing and processing of the touch event chain, which are common to all manager classes, are not represented in the diagrams presented hereafter.

Mode Connector

The Touch Adapter provides access to contact data, which allows developers to define mode-setting postures based on different touch patterns. While such posture definitions can obviously be hard-coded in the corresponding components and gestures that make use of them, such a design choice impacts the flexibility to later change touch shapes associated with specific modes. For instance, if a command mode needs to be changed from a three-finger chord posture to a flat-hand shape, every code instance in which that particular touch configuration is detected needs to be modified. The Mode Connector is a centralised means to link input signatures with semantic associations to specific modes. In the previous example, "command mode" would be represented by a corresponding object or status in the Mode Connector which is associated to a physical posture. Client components wishing to know what the currently active mode can query the Connector, which keeps track of set modes when receiving touch input events.

Figure 3.4 shows the Mode Connector and its relationships with `Mode` and `Posture` objects for which it stores associations. Modes are typically declared collectively as static entities within a public object so that they can be accessed anywhere using their denominations e.g. `Modes.COMMAND`, `Modes.GESTURE` etc. Optionally, rendering properties can be assigned to a `Mode` so that graphic objects associated with it can be painted in a consistent manner. This option is especially meant for pen strokes that should be drawn in a particular way, depending on the active mode. For instance, programmers could define a command mode, with thick red pen strokes, a selection mode with a yellow transparent dashed stroke style etc.

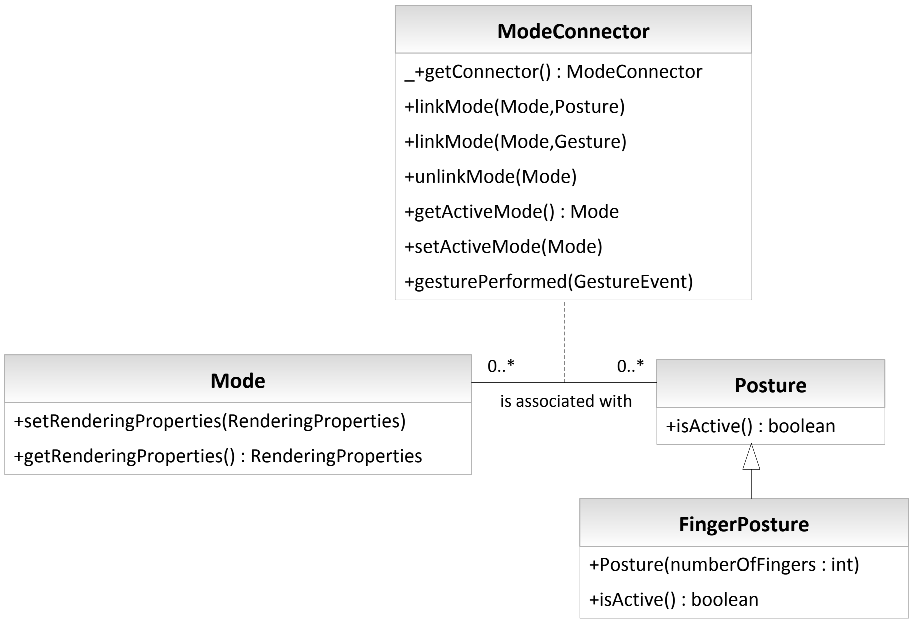


Figure 3.4: The Mode Connector with associated `Mode` and `Posture` objects

A Posture is mainly defined by whether it is active in response to a given touch pattern. Implementations of the `isActive` method return `true` if the conditions for the posture are met. A convenience subclass is provided to easily create chord-based postures. Internally, this class simply queries the Dispatcher's `getNumberOfTouchFingers` method to check if the number matches that required by the posture. Postures characterised by contact shapes analyse the `Shape` object returned by the `getContactShape` function to determine whether it fits the parameters set for the posture. Listing 3.1 shows an example of how a palm posture can be detected.

Postures are instantaneous mode-indicators, that is, their activation is determined based on a single touch frame. Programmers might however wish to set modes based on offline gestures, i.e. after an interaction has been completed. As shown in Figure 3.4, the Mode Indicator provides an additional `linkMode` method to associate a `Mode` with a `Gesture`. Since notification of gesture execution is event-based, mode-setting in this case occurs in the `gesturePerformed` method, which the Mode Connector implements as a registered `GestureListener`. The gesture model is described in the next subsection.

Several Postures or Gestures can be associated to the same mode in the Mode Connector. However, there can be only one active mode at a time. The hashmap that internally stores the association returns only a single `Mode` object upon calling of the `getActiveMode` method and it is the responsibility of the developer to manage potential conflicts between interactions that may occur simultaneously but are associated with different modes. In practice, mode-triggering should realistically be limited to a few finger-based postures and/or one or two non-overlapping shape postures.

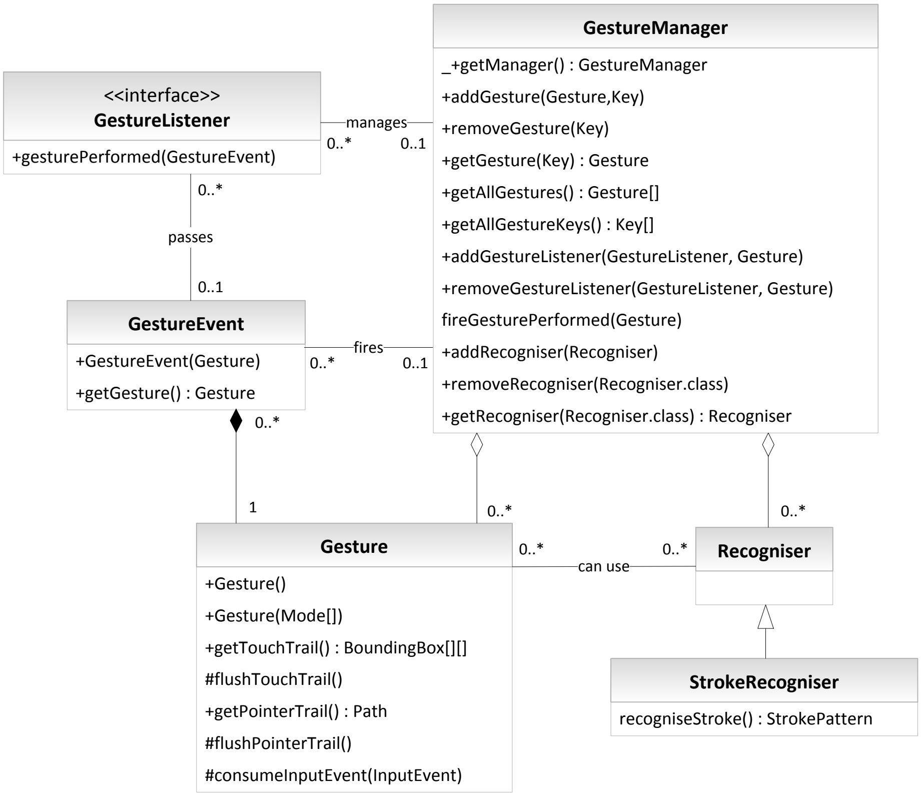
```
public boolean isActive()
 {
  Shape contactShape = InputDispatcher.getContactShape();
  if (contactShape!=null)
   {
    BoundingBox boundingBox = contactShape.getBoundingBox();
    return (boundingBox.width>Settings.palmMinimumWidth &&
            boundingBox.height>Settings.palmMinimumHeight &&
            InputDispatcher.getNumberOfTouchFingers()==0);
   }
  return false;
 }
```

Listing 3.1: Implementation of the `isActive` method for a palm posture. The posture is recognised if the touch bounding box exceeds a certain dimension and the number of detected fingers is 0 (this last condition is necessary because multi-finger interactions such as two-finger zooming may also generate large bounding boxes)

Gesture Manager

The Gesture Manager is the central entity governing gesture processes. It contains a list of all the gestures defined for a particular application to which it routes input data and controls the propagation of gesture events to registered listeners. The Manager also holds a list of recogniser engines that can be used by developers to create new gestures. Gestures, listeners and recognisers can be dynamically added to and removed from the Manager using corresponding methods (Figure 3.5).

Gestures are defined by subclassing the `Gesture` class and overloading the desired input event methods (`touchDown`, `penDown` etc.). Events for both pen and touch input are included, so that programmers can implement unimodal and hybrid gestures. A constructor with an array of `Mode` objects is also available to create gestures that will only receive input events if one of the specified modes is active. Furthermore, the Gesture class provides two methods that record trails of input events for each modality: `getTouchTrail`, which returns an array of arrays of bounding boxes and `getPointerTrail`, which returns a path formed by pointer coordinates (the path need not be continuous). Those functions are useful for clients that require access to the full motion sequence, for instance, to determine affected UI components, which, as we have seen in the introduction, should not necessarily be limited to the element hit upon touch down or pen down. Subclasses are responsible for flushing the input trails when accumulated data needs to be reset.



<<interface>>
GestureListener
+gesturePerformed(GestureEvent)
manages
0..*
0..1
GestureManager
_+getManager() : GestureManager
+addGesture(Gesture,Key)
+removeGesture(Key)
+getGesture(Key) : Gesture
+getAllGestures() : Gesture[]
+getAllGestureKeys() : Key[]
+addGestureListener(GestureListener, Gesture)
+removeGestureListener(GestureListener, Gesture)
fireGesturePerformed(Gesture)
+addRecogniser(Recogniser)
+removeRecogniser(Recogniser.class)
+getRecogniser(Recogniser.class) : Recogniser
0..*
passes
0..1
GestureEvent
+GestureEvent(Gesture)
+getGesture() : Gesture
fires
0..*
0..1
0..*
1
0..*
0..*
Gesture
+Gesture()
+Gesture(Mode[])
+getTouchTrail() : BoundingBox[][]
#flushTouchTrail()
+getPointerTrail() : Path
#flushPointerTrail()
#consumeInputEvent(InputEvent)
0..*
can use
0..*
Recogniser
StrokeRecogniser
recogniseStroke() : StrokePattern

Figure 3.5: The Gesture Model with its constituent elements

For the detection of particular gestural patterns, programmers can avail themselves of recogniser engines, for example, by supplying the path of the pointer trail and checking if the result matches the target pattern. An instance of a particular recogniser engine can be obtained from the Gesture Manager by calling its `getRecogniser` method and specifying its class name. Clients can then cast the object to the corresponding Recogniser engine and call its class-specific methods.

A gesture set can be designed in several ways, depending on the needs of the application. Gestures can be defined individually with different classes for each gesture, or a class can represent a family of gestures of the same type (e.g. single-stroke gestures). Gestures are uniquely identified in the Gesture Manager through a key specified upon registration. Clients, and particularly listeners, can then determine gestures either based on their class name or through their key. Lists containing all gesture objects and all gesture keys can be obtained via shortcut methods, respectively `getAllGestures` and `getAllGestureKeys`.

```
public class RectangleGesture extends Gesture
 {
  Rectangle rectangle;

  // this gesture reacts only on pointer released so this is the only
  // overridden method
  public void pointerReleased(PointerEvent e)
   {
    Path pointerTrail = getPointerTrail();
    Point[] points = pointerTrail.getAllPoints();
    if (points.length>3) // a rectangle needs to have four corners
     {
      SimpleShapeRecogniser simpleShapeRecogniser =
              (SimpleShapeRecogniser)GestureManager.getRecogniser(
                                         SimpleShapeRecogniser.class);
      Object recognisedObject = simpleShapeRecogniser.
                                  recogniseShapeSingleStroke(points);
      if (recognisedObject != null && recognisedObject instanceof
                                                         Rectangle)
       {
        Rectangle rectangle = (Rectangle)recognisedObject;
        // compute rectangle side lengths
        double sideLength1 = MathUtils.distance(rectangle.p1.x,
                    rectangle.p1.y, rectangle.p2.x, rectangle.p2.y);
        double sideLength2 = MathUtils.distance(rectangle.p3.x,
                    rectangle.p3.y, rectangle.p2.x, rectangle.p2.y);
        // only accept rectangles with a minimum side length
        if (sideLength1>Settings.rectangleMinimumSideLength &&
            sideLength2>Settings.rectangleMinimumSideLength)
         GestureManager.getManager().fireGesturePerformed(this);
       }
     }
    flushTouchTrail();
    flushPointerTrail();
   }

  public Rectangle2D.Double getRecognisedRectangle()
   {
    return rectangle;
   }
 }
```

- - - - - - - - - - - - -

```
RectangleGesture rectangleGestureCommand = new
                                             RectangleGesture(Modes.COMMAND);
gestureManager.addGesture(rectangleGestureCommand,"Rectangle Command");
```

Listing 3.2: A possible implementation of a rectangle stroke gesture using a recogniser (top) and adding the gesture to the Gesture Manager (bottom)

Listing 3.2 illustrates the concepts presented above with an example stroke-based gesture, which uses a recogniser on pointer release. The second snippet shows how the gesture is instantiated with a mode requirement (in this case a pre-defined `COMMAND` Mode) and how it is added to the pool of gestures in the Gesture Manager.

The sample gesture in the code example is of the offline type and thus typically fires a "gesture performed" event on touch or pointer released. The gesture framework can however also be used to implement online gestures, where such events can be emitted within `touchDuring` or `pointerMoved` methods, i.e. on a per touch or pointer-frame basis. Of course, direct interactions can also be coded directly inside a component or an associated class. This is the case for classic interactions such as single- and two-finger panning, zooming and rotating, which are built-in features of high-level UI components and explained further below.

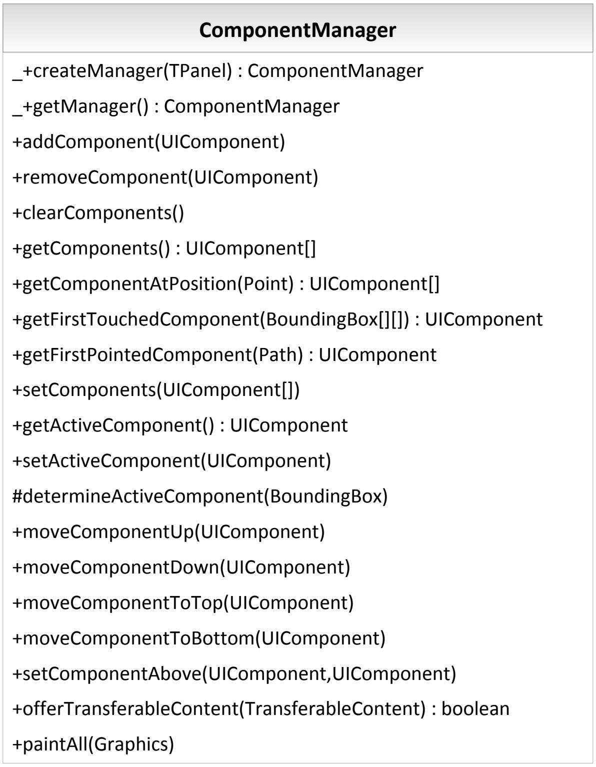
As with mode-setting postures, the programmer is responsible for taking care of potential conflicts between overlapping gestures. Some frameworks attempt to address this issue using priority mechanisms, by allowing higher-priority gestures to execute before and possibly block lower-priority gestures if matched [64, 118, 170]. The problem with such schemes is that priorities cannot always clearly be established simply based on gesture type and input events. The decision to trigger one or another gesture may depend on other factors, such as mode, target component(s), timing etc. Besides, one can argue that it is bad design to have conflicting gestures in the first place. Ideally therefore, the gesture set should be carefully thought out with each gesture gracefully integrating into the whole ensemble. While the framework does not explicitly support priority-based gesture processing, programmers can implicitly achieve such functionality by adding listeners for specific gestures in the order they wish the associated events to be fired. For instance, if square and rectangle stroke gestures are defined (again, a bad design choice), the former can be set to be triggered before the latter by first adding the corresponding listener of the square gesture before that of the rectangle gesture. Thus, rectangle stroke gesture events can be discarded in the `gesturePerformed` method if a square gesture was previously detected and processed.

Component Manager

The third major entity in this layer is the Component Manager, which centralises the administration of UI components and initiates the rendering pipeline (`paintAll` method). The Manager is created using a static factory method to which a `TPanel` object is passed, which is a special component at the root level of the UI hierarchy.

Other than conventional functions to set, add and remove components, the Manager defines an active component, which is, in most cases, the element hit by a touch or a pen down (computed by the internal `determineActiveComponent` method). Clients can furthermore change the layering or z-order of components on the 2D rendering space with the `moveUp`, `moveDown`, `moveToTop`, `moveToBottom` and `setComponentAbove` methods as well as obtain components located at a particular position: `getComponentAtPosition`, `getFirstTouchedComponent` and `getFirstPointedComponent` (the latter two functions are convenience methods used to retrieve target components when processing gestures). Finally, the Component Manager also mediates content transfer operations between components via its `offerTransferableContent` method.

`TPanel`, `UIComponent` and the content transfer mechanism are described in the next subsection.



```
_+createManager(TPanel) : ComponentManager
_+getManager() : ComponentManager
+addComponent(UIComponent)
+removeComponent(UIComponent)
+clearComponents()
+getComponents() : UIComponent[]
+getComponentAtPosition(Point) : UIComponent[]
+getFirstTouchedComponent(BoundingBox[][]) : UIComponent
+getFirstPointedComponent(Path) : UIComponent
+setComponents(UIComponent[])
+getActiveComponent() : UIComponent
+setActiveComponent(UIComponent)
#determineActiveComponent(BoundingBox)
+moveComponentUp(UIComponent)
+moveComponentDown(UIComponent)
+moveComponentToTop(UIComponent)
+moveComponentToBottom(UIComponent)
+setComponentAbove(UIComponent,UIComponent)
+offerTransferableContent(TransferableContent) : boolean
+paintAll(Graphics)
```

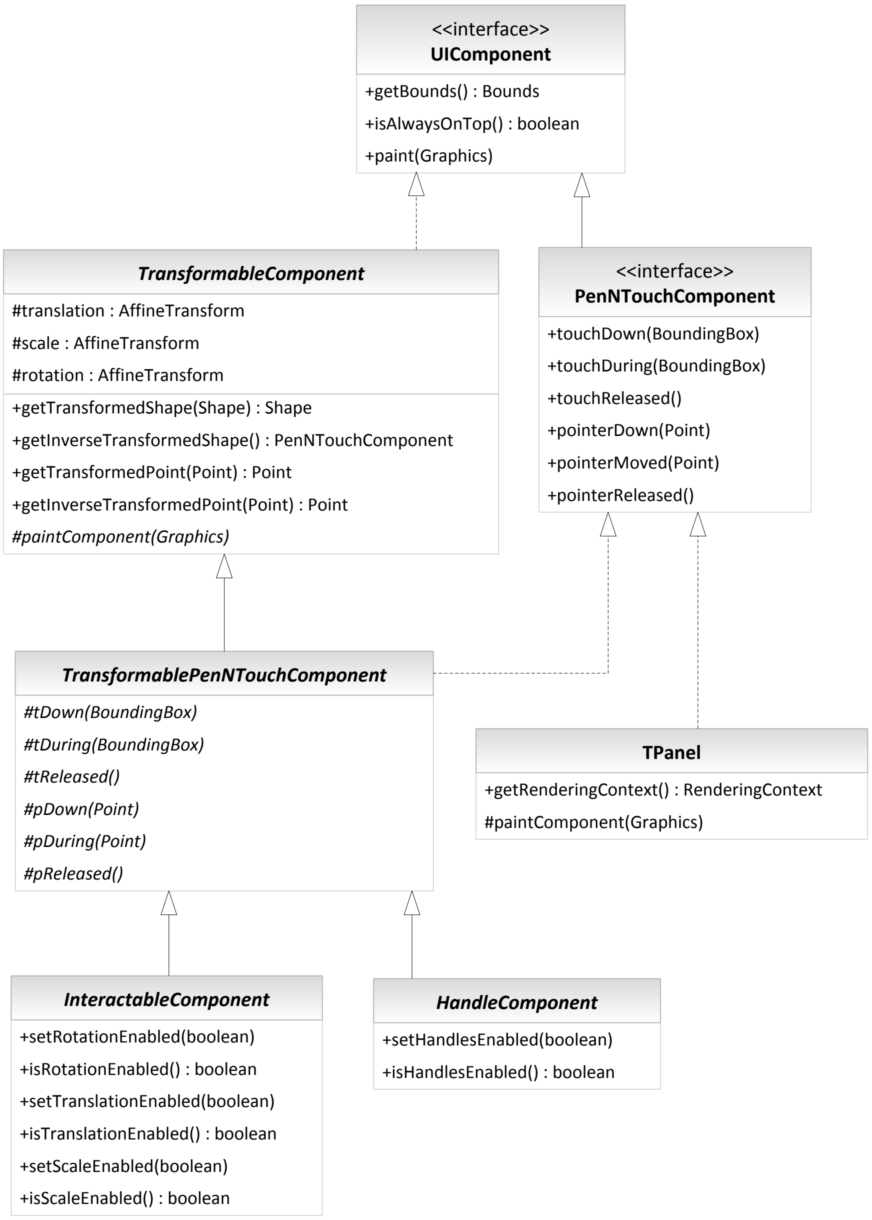
Figure 3.6: The Component Manager

3.2.5 The Component Layer

The UI layer comprises the top-level components of the user interface as well as a module to handle the transparent transfer of user content between components.

Component Hierarchy

Figure 3.7 shows the basic component hierarchy with the main interface characterising a UI component: `UIComponent`. This interface simply defines the signature of methods to render and return the bounds of the component and to specify whether it should always be painted on top of other elements (e.g. for overlays). `UIComponent` is the minimum interface to realise for a valid component to be created and added to the Component Manager (see Figure 3.6). Developers can thus create simple graphical, i.e. passive, components by simply realising the interface and implementing its paint method.



<<interface>>
UIComponent
+getBounds() : Bounds
+isAlwaysOnTop() : boolean
+paint(Graphics)
TransformableComponent
#translation : AffineTransform
#scale : AffineTransform
#rotation : AffineTransform
+getTransformedShape(Shape) : Shape
+getInverseTransformedShape() : PenNTouchComponent
+getTransformedPoint(Point) : Point
+getInverseTransformedPoint(Point) : Point
#paintComponent(Graphics)
<<interface>>
PenNTouchComponent
+touchDown(BoundingBox)
+touchDuring(BoundingBox)
+touchReleased()
+pointerDown(Point)
+pointerMoved(Point)
+pointerReleased()
TransformablePenNTouchComponent
#tDown(BoundingBox)
#tDuring(BoundingBox)
#tReleased()
#pDown(Point)
#pDuring(Point)
#pReleased()
TPanel
+getRenderingContext() : RenderingContext
#paintComponent(Graphics)
InteractableComponent
+setRotationEnabled(boolean)
+isRotationEnabled() : boolean
+setTranslationEnabled(boolean)
+isTranslationEnabled() : boolean
+setScaleEnabled(boolean)
+isScaleEnabled() : boolean
HandleComponent
+setHandlesEnabled(boolean)
+isHandlesEnabled() : boolean

Figure 3.7: The basic component hierarchy

Immediately subclassing `UIComponent` is the `PenNTouchComponent` interface, which adds methods to react on input data. When processing rendering updates and input events, the Component Manager is able to detect the component's class and route events accordingly so that programmers can use the same channel for both passive and interactive components. Specifically, the Component Manager will call a component's `touchDown`, `pointerDown` etc. methods if present.

The other subclass of `UIComponent` is the abstract class `TransformableComponent`, which defines a component to which the three classic affine transformations can be applied: translation, rotation and scale. Those transformations are executed on the `Graphics` object passed in the `paint` method that the component realises before internally calling the abstract `paintComponent` method that subclasses should implement. The component provides corresponding methods to convert coordinates of points and shapes from global space to component space and vice versa.

Extending `TransformableComponent` and realising the `PenNTouchComponent` interface is the `TransformablePenNTouchComponent` abstract class, which further declares a set of event processing methods matching those of `PenNTouchComponent`: `tDown`, `tDuring`, `tReleased`. Those methods, exposed to subclasses for implementations of component-specific behaviour, are called by `touchDown`, `touchDuring` and `touchReleased` respectively, after the affine transformations responsible for panning, scaling and rotating have been given a chance to update based on the input events received through the `PenNTouchComponent` interface. The framework provides two subclasses implementing two common ways of associating touch interactions to such transformations: the `InteractableComponent`, which integrates the classic two-finger pinch/spread zoom and rotate interaction pattern and the `HandleComponent`, which performs resizing operations via control handles located around the component (see Figure 4.2 for illustrations). Hence, programmers who would like to create custom components incorporating one or the other manipulation method need only extend the desired class and implement the `tDown`, `pDown`, `tDuring` etc. methods. `InteractableComponent` and `HandleComponent` include means to respectively activate and deactivate individual transformations and enable or disable control handles.

Finally, at the top level of the component hierarchy is the `TPanel`, a special `PenNTouchComponent` which represents the tabletop screen on which other components are displayed. The `TPanel` is only created once and is passed as a parameter to the Component Manager, which adds it at the top of its internal data structure. The class provides a method to obtain the rendering context of the application, so that information such as screen dimensions and rendering properties can be accessed. A `paintComponent` method is also available for subclasses to draw custom content (e.g. change the background colour, show debugging information etc.).

Listing 3.3 shows an implementation of a canvas component, which assigns four different functions to the pen, depending on the activated mode: inking (normal), delete, select and gesture. On `pointerDown`, the currently active mode is acquired from the Mode Connector and used to instantiate a `ModedPenStroke`, an object that encapsulates a 2D path and whose rendering style is derived from the Mode's associated `RenderingProperties` object (see Figure 3.4). As the user draws on the component with the pen, points are added to the Stroke object, which is rendered accordingly in the `paintComponent` method. The example illustrates two typical ways in which mode-dependent operations can be carried out following a completed pen stroke. The first three are called directly in the `pointerReleased` method, whereas the fourth relies on a Gesture and hence is performed inside the `gesturePerformed` method, defined in the `GestureListener` interface. The example provided for the latter case is that of a `CrossGesture`, which, if detected on the target component, clears all drawn strokes.

Because the Canvas extends `InteractableComponent`, it is automatically endowed with its interaction capabilities allowing panning, zooming and rotating with two-finger gestures. Further extensions of `ModeCanvas` with additional functionality can be built by subclassing the component and implementing the `tDown`, `pDown` etc. methods.

```
public class ModeCanvas extends InteractableComponent implements
                                                    GestureListener
 {
  Vector<ModedPenStroke> penStrokes = new Vector<ModedPenStroke>();
  ModedPenStroke currentPenStroke;
  Mode startMode;

  public void pointerDown(PointerEvent pe)
   {
    ModeConnector modeConnector = ModeConnector.getConnector();
    startMode = modeConnector.getActiveMode();
    currentPenStroke = new ModedPenStroke(currentMode);
    // transform point to component coordinate system first
    Point transformedPoint = getInverseTransformedPoint(pe.getPoint());
    currentPenStroke.addPoint(transformedPoint);
    // call pDown method for subclasses
    pDown(pe);
   }

  public void pointerMoved(PointerEvent pe)
   {
    ModeConnector modeConnector = ModeConnector.getConnector();
    // cancel action if user switched mode during pen tracing
    if (modeConnector.getActiveMode != startMode) return;
    Point transformedPoint = getInverseTransformedPoint(pe.getPoint());
    currentPenStroke.addPoint(transformedPoint);
    pDuring(pe);
   }

  public void pointerReleased()
   {
    ModeConnector modeConnector = ModeConnector.getConnector();
    if (modeConnector.getActiveMode != startMode) return;
    // perform action depending on current mode
    if (startMode == Modes.NORMAL)
     penStrokes.add(currentPenStroke);
    else if (startMode == Modes.DELETE)
```

```
      deleteStrokes();
     else if (startMode == Modes.SELECT)
      selectStrokes();
     pReleased();
     currentPenStroke=null;
    }

   protected void paintComponent(Graphics graphics)
    {
     for (ModedPenStroke penStroke : penStrokes)
      penStroke.paint(graphics);
     if (currentPenStroke != null)
      currentPaintStroke.paint(graphics);
    }

   public void gesturePerformed(GestureEvent ge)
    {
     Gesture gesture = ge.getGesture();
     // get component first touched by this gesture
     UIComponent component = ComponentManager.getManager().
                getFirstTouchedComponent(gesture.getTouchTrail());
     // check if gesture was performed on this component
     if (component == this)
      {
       if (gesture instanceof CrossGesture)
        {
         // delete all strokes
         penStrokes.clear();
        }
       // add other gestures if needed
      }
    }


   // delete strokes (selected group or intersected)
   private void deleteStrokes()
    {
     ...
    }

   // lasso-select drawn strokes
   private void selectStrokes()
    {
     ...
    }

  }
```

Listing 3.3: Code sample for a drawing canvas component with inherited navigation behaviour and non-inking modes to delete and select strokes (implementations for those methods are left out)

Transparent Content Transfer

A common requirement in pointer-based and multitouch tabletop interfaces is to copy or move content from one location to another, where this action may also have a functional meaning. In many cases this action is performed by drag and drop, but other types of interactions can be envisaged for content transfer, e.g. touch source, touch target, touch hold + pen drag etc. It is up to the developer to define the transfer conditions. To ensure the independence and decoupling of components in the internal software structure, source and target containers should not be hard-linked to each other, unless, of course, there is a specific requirement to do so.

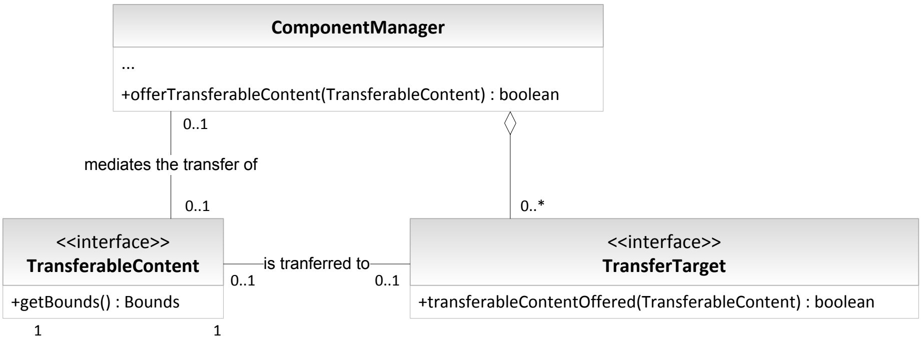


Figure 3.8: The entities involved in the content transfer mechanism

The framework integrates a set of interfaces, which are conceptually similar to those found for drag and drop in other software platforms, to facilitate content transfer between sources and targets. The relevant entities are shown in Figure 3.8.

To indicate that a `TransferableContent` is available for transfer, the initiating component calls the `offerTransferableContent` method of the Component Manager, which scans for `TransferTarget` components in its internal list. When such components are found, their `transferableContentOffered` method is called. If the `TransferTarget` accepts the transferred object, `offerTransferableContent` returns true to signal this to the source. If all Targets reject the transferred content or if the Component Manager has no registered `TransferTarget`, `offerTransferableContent` returns false so that the initiating component may respond accordingly.

3.3 Implementation

3.3.1 Hardware and I/O

As enabling hardware for multitouch sensing and display, a DiamondTouch DT107 with a 1600×1200 projector was used (yielding a dpi value of roughly 48). For the styli, DP-201 and ADP-301 Anoto pens were utilised. The former is designed for offline work and synchronises via Bluetooth or a dock. However, thanks to the iServer framework [175], which contains a module that directly reads the pen's raw input data, streaming information can be captured, which makes online use possible. The price to pay is a certain degree of latency. The newer ADP-3010 pens, on the other hand, explicitly support streaming and have a reduced lag.

Sensing data for those pens is obtained through an SDK provided by the Media Interaction Lab of the University of Applied Sciences Upper Austria [7].

3.3.2 Software Platform

Most of the framework was implemented in Java SE 6 on Windows XP, with only a few native components at the lower levels (mainly for I/O management). Implementations of recognisers integrated in the Gesture Manager include $1 [205] and an adapted version of Jorge and Fonseca's algorithm [102] with custom code to derive clean geometrical shapes from the stroke data.

3.4 Summary

This chapter described the software framework created for the purpose of facilitating development of application prototypes on pen and touch-operated tabletop devices. At the raw input level, the design effort attempted to abstract raw input data taking into account the specific characteristics of the sensing hardware and its limitations, in particular those of the DiamondTouch. For multitouch, a viable common denominator for devices able to detect shape contacts and the DiamondTouch was determined, based on anticipated application requirements. The platform-independent input data abstraction thus determined consists of the global bounding box of touch contacts and the number of fingers touching the surface. Part of the motivation for this choice was to explicitly support NDH-driven pen mode switching using shape and chord-based postures. As for pen input, the generic pointer data object includes coordinates and pressure, with provisions to add further properties such as tilt and azimuth angles.

At a higher level, the framework integrates a set of adaptable structures to create and manage UI components and their associated interactions. The central entity responsible for managing components offers a number of functions for clients to add, access, remove and control the z-order of components. The included gesture module allows developers to design offline and online, unimodal and hybrid gestures, with the help of multiple recogniser engines. Mode constraints can be easily added to defined gestures to restrict their execution and recognition context. Code for a sample gesture illustrating the introduced concepts was provided. As for the components themselves, a flexible hierarchy was presented with ready-made components integrating two standard interaction patterns for basic navigational manipulations. An example implementation of a canvas component, which supports different mode-dependent pen actions, was given. Finally, a mechanism to transfer user content between components was described.

In the next chapters, a series of prototypes developed using this framework as a foundational base are presented. While those systems will be mostly described in a functional and HCI perspective, details about how the framework was used to implement components and features will also be given.

The following chapter reports on a four-part study that was conducted to address some of the contentious or unresolved issues raised in Chapter 2 as well as other important aspects of pen and touch, such as palm resting.

4 Empirical Evaluation of Pen & Touch Interaction Properties

This chapter reports on an empirical investigation of certain important properties of pen and touch interaction in different conditions through a set of four user experiments. From the results of these trials a number of observations and lessons are drawn, with a view to informing the design of applications based on this interactive ecosystem. The first set of experiments tackle issues that have been studied for touch and other pointing devices. Those tests are revisited and adapted for the context of pen and touch tabletop interfaces. Specifically, the tasks involve widget targeting and positioning as well as shape-tracing, where direct (multi)touch input is compared to pen and touch configurations. The difference to previous comparative studies is that, in all pen conditions, the copresence of touch is assumed, either for assisting manipulations as in the first experiment (panning and zooming the workspace), or as a possible hindrance as in the second task (palm resting causing interference). The last two experiments address problems more specific to asymmetric pen and touch interaction: pen-mode switching and bimanual coordination. Different methods to change the operational context of the pen with the NDH are considered: toolbars, in-place radial menus and, for the fourth task, chord-based postures. In the first and third cases, mode changing via short contact actions (tap) or maintained activations (quasimode) is also compared.

The aspects covered in these studies certainly do not encompass the whole problematics of two-handed pen and touch input, but they form an important empirical foundation for the purpose of this project, and presumably, by virtue of their relative generality, for any system based on this interaction model as well.

Part of the material of this chapter was previously published in the proceedings of the ITS 2012 conference [140].

4.1 Related Studies

As has been seen in the previous chapters, the properties of pen and touch interaction have mainly been derived from extrapolations of studies on bimanual interaction and researchers' intuitions as to how best to leverage this interaction paradigm. Many of those aspects have not been empirically confirmed in the pen and touch context, although a number of high-level user evaluations have been conducted [85, 93, 213]. An exception is perhaps Brandl et al.'s maze experiment, in which participants' ability to navigate through a maze with 3 different assignments of tools for the two hands is assessed, specifically pen+pen, pen+touch and touch+touch [44]. The task consists in tracing a path through a maze avoiding walls and using the NDH to pan and zoom the interface to switch between close-ups for increased precision and bird's eye views for route planning. The authors report that pen+touch is superior to bimanual/ unimodal input combinations in terms of speed, accuracy and user preference, although the results for pen+touch and pen+pen are much closer. The decision to include a means to control the view with the NDH is interesting, because it departs from typically rigid target acquisition and trajectory-based tasks, which are often related to Fitts' law [70]. The authors make the very valid point that in many multitouch applications, panning and zooming are standard features users expect to have at their disposal when performing detailed work and so including this functionality in experiments provides better ecological validity.

While empirical research investigating the properties of bimanual pen and touch interaction is still relatively scarce, there have been a number of studies dedicated to mice and other pointing devices [72, 113, 215]. Other evaluations looking at different and potentially competing pointing methods and devices have been conducted, but most of them are mode-comparative by design and so do not examine the interplay of those input techniques when coexisting in a single cohesive environment [58, 132, 191, 212]. Nevertheless, those studies are instructive, not only because their results can sometimes be extrapolated or generalised to other ecosystems, but also because one can draw upon their methodologies. Thus, the designs of the first two experiments presented hereafter were respectively informed by Forline's et al.'s comparison of unimanual and bimanual touch vs. mouse on a tabletop display [72] and to a lesser extent Zabramski's evaluation of mouse, pen and touch input in the context of a freehand shape-tracing task [212].

4.2 Study design and environment

For the design of the evaluation and its constituent tasks, a low to mid-level approach was adopted. This strategy was motivated by the desire to integrate both fine-grained per-action and per-pointer indicators, such as target pointing and tracing errors, as well as higher-level aspects pertaining to user interface elements such as mode-switching and two-handed shortcuts for the design of the tasks. As suggested by Brandl et al. [44], it is hard to generalise and claim ecological validity for measurements obtained from strictly low-level trials that focus on isolated input techniques in a heavily controlled and constrained interactive context. Similarly, it is difficult in higher-level experimental settings to validate individual design choices for more complex applications comprising several features and interaction capabilities. The aim for these experiments, therefore, was to strike a compromise between those two considerations in order to obtain meaningful results that can be of use.

Following the above rationale, a set of four relatively simple tasks were devised. To reduce bias towards a particular category of users, the trials were designed so that they could be easily executed both by people familiar with touch interfaces and those with less or even no experience with those devices. For all four tasks, three different interaction contexts, hereafter referred to as configurations, were created. These configurations represent the three levels of the independent variable that is considered in each experiment (in each case, a particular interaction method).

4.2.1 Study Design

A between-subjects design was chosen for the experiments, as people could presumably become confused if they had to successively perform the same tasks using different interaction techniques. While imposing a logistic burden regarding the number of participants to be recruited, this choice allowed to have volunteers carry out all three experiments in one session, where each person executed one task set to one particular configuration. This design also permitted the use of identical protocols for each configuration, hence direct comparisons were possible without randomising the particulars or orders of the subtasks, which would otherwise have been required to mitigate learning effects.

Before each session, the system was carefully calibrated in order to guarantee the accuracy of the measurements.

4.2.2 Participants

30 participants - 19 males and 11 females aged between 20 and 65 years old (median: 28 years old) - were recruited among students and staff of the department to take part in the study. Three people were left-handed, which justified adopting a symmetrical design for the interfaces (UI widgets and shapes were mirrored about the middle vertical axis for those users).

The overwhelming majority of participants owned one or more touch device(s) such as a modern smartphone and/or a tablet computer, with only four people declaring that they did not possess such machines. No one owned a stylus-operated appliance, except for one user who said he occasionally used a special pen to write notes on his touch tablet.

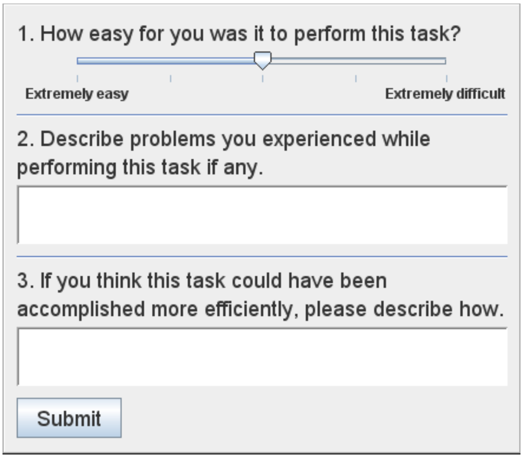
In order to identify the "power users", participants with touch devices were also asked to indicate how much time they spent in a week engaged in touch-intensive tasks, i.e. activities that require continuous interaction and concentration such as typing, active web browsing, games etc. Here, 9 people said they dedicated between 2 and 5 hours to such activities, 7 between 30 minutes and 2 hours, 5 between 5 and 10 hours and 2 users a maximum of 30 minutes.

Since the tasks had three configurations, an even assignment of those configurations to participants yielded three groups of 10 users. Before executing the actual task of each experiment, for which measurements and observations were recorded, participants were given as much practice time as they desired to become acquainted with the work to be done and the different gestures to accomplish it.

For all tasks, the completion time was recorded as well as the time taken to execute each individual subtask. A task constituted of 20 subtasks in each case, except for Experiment 3, which had 30 subtasks. This amounted to 200 subtasks per group for Experiments 1, 2 and 4 and 300 subtasks for Experiment 3.

For each experiment, a number of task-dependent metrics were also logged. Those metrics are described below, in the relevant sections. The obtained values were then summed up for all subtasks to yield a total that was included in the data set used to compare the different configurations in the analysis phase.

After each trial, users were presented with an on-screen questionnaire, in which they had to rate, on a linear scale (using a slider), how easy it was for them to execute the task and report any problems they had encountered (Figure 4.1).


1. How easy for you was it to perform this task?

Extremely easy — Extremely difficult

2. Describe problems you experienced while performing this task if any.

3. If you think this task could have been accomplished more efficiently, please describe how.

Figure 4.1: The common user feedback panel asking users to rate their experience at the end of each task

4.3 Experiment 1

4.3.1 Description

Motivation

One of the essential interactions in applications involving floating UI objects (be those widgets or user content) is the ability to move and place those objects efficiently and precisely. In a rich document editor for instance, document elements such as images, text boxes, cliparts etc. need to be placed in specific locations. While advanced publishing software has particular tools to deal with positioning, alignment and content distribution, they disrupt the natural flow of the user's movement and are not always tuned to their liking (e.g. snap-to-grid competing with snap-to-object or constraint thresholds not matching the granularity of the user's current operation). It seems therefore worth investigating how positioning objects can best be done using pen and touch to estimate what kind of raw unaided precision can be achieved.

Another aspect of element positioning that is important and yet not considered by the majority of related studies in the literature, is the influence of the finger or pointer lifting movement when releasing the object. Indeed, the action of positioning an object does not end with its placement at the desired location on the surface, while the finger is still holding it. To finalise the operation, the user needs to disengage the object, thereby possibly causing it to further move slightly as the contact area is modified upon lifting the finger. This undesired effect, hereafter referred to as the displacement-on-release effect, is often observed in interactions involving the precise positioning of a cursor, for instance, when scrubbing to place the slider of a media player at a specific time position. In situations where high precision or reactivity is not important, smoothing or thresholding filters can be used to counteract this effect (those filters are incidentally used for anti-jittering when the finger is held down on the screen and the system needs to discard slight changes in the contact area). In [195], Wang and Ren measure the distances between what they call stable and "lift up" finger states, among various other finger touch properties. The stable state is defined as the moment when the contact area is largest and the lift up state as the last registered contact shape. For the index, they obtain an average deviation of 1.37 mm with an oblique touch, the highest value of all fingers. The tasks used by the authors only involved direct tapping and rolling of fingers and there were no positioning or docking actions, which might have some influence on the results. Moreover, since they only studied finger touch properties, there is no comparison with pens. Measuring the displacement effect in a docking task with pen and touch should therefore yield interesting –and more ecologically valid– data, which can then be compared to the 1.37 mm value.

Design

As mentioned above, this experiment was inspired by Forlines et al.'s study of direct-touch vs. mouse input [72], which somewhat departs from a traditional Fitts' law-driven target acquisition task. The experiment involved docking a draggable rectangular shape inside a given target, where the rectangles had to be both moved and resized, as the two actions are often available together in applications with those types of operations. In effect, this corresponded to a merging of Forlines et al.'s two experiments. Additionally, a virtual sheet in the shape of a blank document (hereafter referred to as "the worksheet") in the middle of the workspace was included to mimic a real work environment in an editing application. This worksheet could be panned and zoomed using classic multitouch gestures, facilities that are commonly available in such kind of interfaces, as has been mentioned previously, and whose influence on the docking task was interesting to see. All shapes appeared and were to be manipulated on this designated space. Users could also drag and release the rectangles as many times as they wished.

As soon as an edge was positioned within 2 pixels (with a worksheet scaling factor of 1) of the corresponding target edge, its colour changed to green (Figure 4.2). Contrary to Forlines et al's experiments, shapes could only be "validated" after fully releasing them, in order to elicit the displacement-on-release effect. For the implementation of this experiment, the displacement factor was computed slightly differently from Wang and Ren's method, as the intention here was to capture the global movement of the finger when it is lifted. Hence, the displacement was defined as the distance covered by the contact points registered by the system *t* milliseconds before the finger or the pen was lifted from the surface. A value of 200ms for *t* was chosen and this displacement was recorded only for release events that led to a correctly positioned shape in order to ensure that the user was finalising a docking action. An equal number of contact events were considered to account for the sampling rate differences between the two digitising technologies for pen and touch.

In the first configuration, the rectangles had to be moved and resized using two-finger pinch and spread gestures, where the changes of the bounding box formed by the two contact points of the fingers were mapped to the affine transformations affecting the position and size of the rectangle to be dragged in place (Figure 4.2 left). Hence, the gesture commanded translation as well as the two independent scale factors of each axis of the dragged shape. It was left to the user whether to use fingers from the same hand or from both hands to perform the manipulations. Pinch and spread interactions are mostly used for coarse zooming rather than precise scaling of objects but since they are widely established gestures on multitouch devices, it was interesting to see how people could apply them for positioning operations.

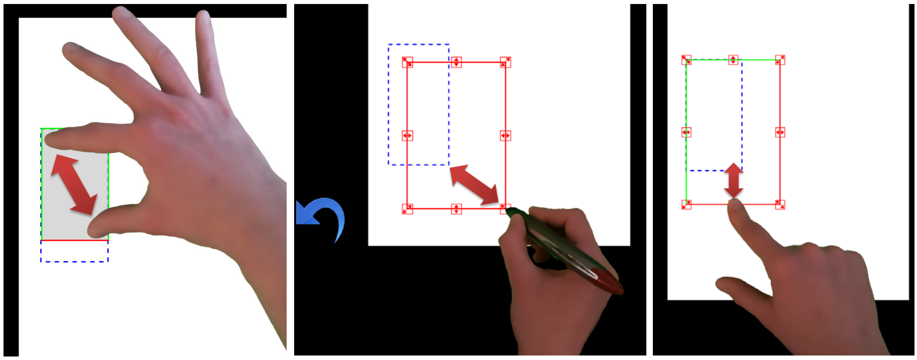


Figure 4.2: Positioning task with pinch-spread moving and scaling (left), unimodal handle manipulation with the pen (middle) and fingers (right)

One of the most common methods to adjust the position and size of elements in graphics and publishing software is via their edges and corners, very often using manipulation aids such as draggable controls. Hence, in the second and third configurations, rectangles were supplemented with handles along their edges and corners. Those handles could be tapped and dragged with the pen (second configuration, Figure 4.2 middle) or finger (third configuration, Figure 4.2 right) to move the associated edges thereby scaling the rectangle. In order to test the influence of the handle size, the dimensions were varied from shape to shape using three different set sizes: small (15px), middle (25px) and large (35px), which roughly represent sizes below, close to and greater than the dimensions of the typical contact bounding box of a finger. The size of the handles scaled along with all objects contained in the worksheet in accordance to the latter's zooming factor. This meant that users could use the scaling feature of the worksheet to increase their targeting accuracy if they wished (but possibly at the cost of more time required to complete the task). Again, this is a departure from similar studies, which do not consider adjustments to the workspace to modify the interaction granularity. Each set handle size was used an equal number of times for the task, specifically 6 times each for a total of 18 rectangles to be positioned.

For the configurations with handles, the number of times users missed the latter was recorded. Recalling that the target scenario was an editing or design program on a pen and touch system, the pen was mainly to be used for inking and direct stroke input. Therefore, in the second configuration where the pen could also be used for handle manipulations, a miss caused a stroke to appear on the worksheet. To be able to resume the positioning operation, users had to tap an undo button located at the bottom left (or bottom right for left-handed people) of the work-space (Figure 4.2, middle). For the third configuration, where handles were manipulated by the finger, a missed handle only caused the user to move the worksheet instead of the object, which was considered to be of lesser impact than accidental inking. Therefore, in this condition, an error was only registered, without imposing additional time-consuming penalties. Because it was difficult to algorithmically differentiate an intentional worksheet pan from a handle miss, users were simply observed and errors were manually recorded when it was obvious they were off target when intending to select a handle. An alternative method would have been to introduce explicit mode switches for the pen so that when it is in non-inking mode, there is no chance of marking the worksheet. However, this would have required a complete redesign of the tasks, as they would have had to include interleaved inking vs. moving actions in order to trigger the mode changes. In fact, mode switching between inking and non-inking modes is studied in Experiment 3, described further below.

4.3.2 Results

Figure 4.3 shows task completion times and difficulty ratings chosen by users after performing the test. The charts show a similar pattern, with the two-finger positioning configuration (Pinch/Spread) appearing to be the most inefficient and difficult.

ANOVAs performed on these results confirmed there were significant effects in both cases ($F_{2,27} = 6.41$, $p = 0.005$ for completion time and $F_{2,27} = 14.34$, $p < 0.001$) so post-hoc Tukey tests were conducted. Those tests revealed significant mean differences for completion time between Pinch/Spread and the two others ($p < 0.001$ in both cases), however no main effects were observed between the two handle configurations ($p = 0.49$). Similar conclusions could be drawn with respect to the difficulty ratings. Hence, it was not possible to directly confirm that pen was significantly faster than touch, as studies like [58] found.

Digging deeper into the results and relating them to observations of participants executing the task, especially the last docking operation was difficult for users as it required precise positioning of the rectangle edges. The displacement-on-release effect caused the rectangle position controlled by the held down finger(s) or pen to slightly move when lifting it/them up, leading to correctly positioned rectangles being misaligned after release. The average value for this displacement was evaluated at 1.18 pixels (0.6 mm) for the pen (SD=0.44) and 2.26 pixels (1.2 mm) for single-finger touch (SD=0.54), which are significantly different values, as confirmed by a t-test ($p<0.001$). 1.2 mm is consistent with Wang and Ren's 1.37 mm and tends to show that the docking action in this case has no major influence. After observing users, that result was not surprising, considering they carefully made sure the rectangles were correctly placed before releasing them. In fact, to avoid unwanted displacements when removing their fingers, some participants, at some point, tried to lift them in a swift vertical motion so that the tabletop sensors did not have time to register significant changes in the contact shape upon release.

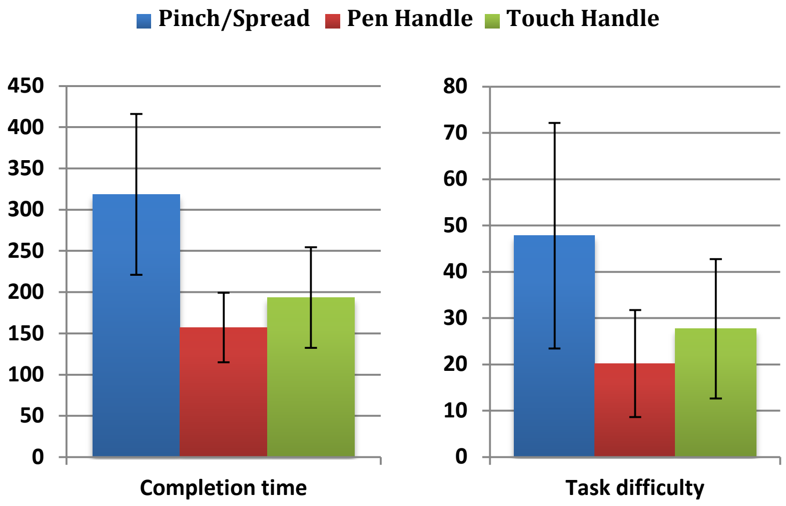


Figure 4.3: Task completion times (in sec.) and user ratings of task difficulty (from 0=extremely easy to 100=extremely difficult). Error bars show standard deviation values.

The reduced accuracy of touch and multitouch translated in increased uses of the worksheet’s zooming feature. All participants of Pinch/Spread made use of it, with an average of 7.4 operations per user (SD=4.48), while 7 did for Touch Handle with 5.1 zoom actions on average (SD=4.1) and only 2 people for Pen Handle.

Observations also showed that users tended to favour sequencing simple actions and separating degrees of freedom rather than attempting to perform one complex operation controlling several factors at a time. Many participants of the "Pinch/Spread" group said in their feedback that they would have liked to have been able to "fix" correctly placed edges so that they could then have concentrated on working with the others. The configurations with single-point interaction with handles controlling only one or two edges fulfilled that requirement and users were more efficient in those conditions and hence felt the task was easier. Those observations are consistent with findings on control allocation of degrees of freedom in 3D manipulation [136].

Turning to the comparison of the handle configurations, specifically the handle targeting errors, an average of 1.6 errors for the pen (SD=1.84) and 6.2 errors for touch (SD=3.61) were recorded, again with significance easily confirmed by a t-test ($p < 0.001$), which corroborates previous studies reporting that pen or stylus is more accurate than touch [3]. As expected, most errors occurred for the smallest handles (68% and 80% respectively) and edges (25% and 13%). As for positioning strategies adopted by participants, no particular differences were observed, as the numbers of move and resize operations were roughly equal for both configurations (38.2 vs. 39.8 and 13.5 vs. 16 respectively).

In a qualitative feedback session, participants of the pen handle group were asked if they would have preferred to use touch and vice-versa, but generally, users of each group did not express any dissatisfaction with their assigned interaction method. Only two people in the Touch Handle condition said that a pen would perhaps have been better to hit the smaller handles, but this was less a problem of touch than a problem of the interface (and a deliberate design choice for the experiment).

Essentially five lessons from this experiment can be derived (new findings are listed in italic, while results echoing prior work remain in plain text):

1. The pinch-spread gesture is not particularly suitable for precise positioning and hence is best reserved for coarse zooming functions of browsing or viewing interfaces with a single common scale for both axes.
2. The pen can also be an appropriate tool for widget manipulations, both in terms of precision/efficiency and the perspective of (some) users.
3. *The pen is not necessarily faster than touch to manipulate objects via control handles, provided those have an adequate size (this is in contradiction with results of prior work).*
4. A succession of simple operations that each controls a limited amount of degrees of freedom is preferred to one complex action that attempts to fulfil multiple operational requirements simultaneously.
5. The displacement-on-release was measured at 1.22 mm for single-finger touch *and 0.6 mm for the pen*. To achieve precise positioning with touch, therefore, this effect needs to be suppressed with adequate filtering and smoothing techniques. Other assisting functionality such as snap features (a suggestion made by a few participants, but contrary to the intent of this experiment), rulers and other constraint-based mechanisms [77, 200] can also be utilised.

4.4 Experiment 2

4.4.1 Description

Motivation

The default function of the pen when editing is most likely going to be inking and while for some scenarios fidelity and tracing precision do not necessarily have a major impact (for instance, for coarse doodling or freehand note-taking without subsequent character recognition), there are many cases where those factors considerably matter, including drawing, CAD, handwriting text to be recognised etc. Stylus accuracy is also important in some non-inking contexts, e.g. when performing lasso selections or alignment operations [77]. For touch-only devices, drawing and tracing tasks otherwise traditionally performed with the pen on paper have to be done with the finger. While most people might have felt uncomfortable writing with fingers prior to the tablet era, this sentiment has now changed, as tablet users have grown accustomed to using touch for all sorts of tasks, including writing and drawing.

Apart from Zabramski's study [212], there is little experimental data comparing pen and touch in tracing tasks. As the researcher points out, shape-tracing is different from navigational tasks, which have been extensively used to verify predictive models such as Fitt's law or the Steering law [25]. The results of his evaluations support the intuitive assumption that people are less accurate with touch than with a pen. His design, however, does not impose temporal constraints, which introduces a subjective bias into the results, as users are left to decide whether to favour accuracy over speed or the opposite. To control for that, one can force people to execute a tracing action within a specific amount of time. But that kind of constraint is also very artificial and there are few real-world examples of pen tasks with specific time limitations outside of games. Hence both types of designs have advantages and disadvantages.

Another important factor for pen tracing that was touched upon in the introduction is the influence of palm resting. Most bimodal systems still do not support robust localised palm rejection, which means users have to draw and write while lifting their arm. To date, there is no study that investigates whether not allowing the palm or the arm to rest on the tabletop surface has a noticeable influence on precision. Naturally, one would surmise that imposing such a constraint should have a negative impact, but by how much?

Design

The above issues motivated the design of the second experiment with the following three configurations: finger tracing, pen tracing with and without palm resting (Figure 4.4). Simple geometrical shapes were used for the tests to allow participants to easily execute single-stroke tracing movements with minimal cognitive effort. Those shapes included smooth, round paths such as circles and ellipses and "hard" shapes with corners such as triangles and rectangles, as it is known that the curvature has an influence on the tracing motion [152, 212]. As in the first experiment, the shapes to be drawn appeared on a document-like virtual worksheet, which this time was static, i.e. it could not be panned or zoomed, in order to reliably deal with the palm-resting issue. The task for the participants consisted of trying to trace over the given shapes in one stroke as precisely as possible. In the third configuration (pen tracing without palm resting), if the user touched the surface with their hand or arm, the background colour would change to red and tracing with the pen would be blocked until the hand or arm was lifted.

As explained above, there is a tension between allowing users to trace over shapes freely without any time pressure, but with a speed-accuracy bias, and imposing temporal constraints but at the cost of diminished ecological validity. In order to consider both cases, time limitations were added only for some of the shapes. To provide users with appropriate awareness of this time pressure without disrupting the tracing task, the passing time was materialised as a light grey area gradually filling the worksheet from top to bottom (Figure 4.4 right). A time constraint was constructed as a function of the perimeter of the shape to be drawn, a fixed offset and a modifier. Three time settings were created for the task: unlimited (i.e. no time constraint, users could take as much time as they wanted to draw over the shape), moderate and fast. Typical times for the fast setting would be less than 1.8 seconds, so users were forced to draw the shapes very rapidly, presumably with a significantly reduced ability to concentrate on precise tracing. The moderate parameter corresponded to times between 4 and 7 seconds, depending on the size of the shape to be drawn. Those two time constraints were attempts to roughly model rapid and moderately rapid sketching or handwriting situations.

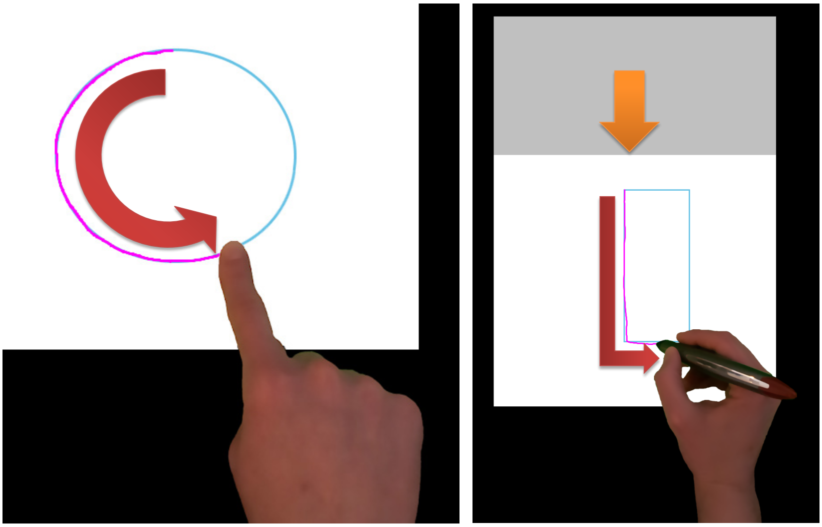


Figure 4.4: Shape-tracing task with a finger (left) and with the pen (right). The grey filler shows the passing time.

A series of 18 different shapes was thus generated for all configurations and, as in the first experiment, the three time settings were evenly assigned to the individual subtasks. Shapes that a user could not finish drawing within their associated time constraints (i.e. a timeout occurred while the pen was down) were repeated at the end of the set, in order to guarantee the consistency of the recorded data.

Two precision errors were calculated for this task: a pointer-down error, which was the distance from the first contact point to the closest point on the target shape, and a shape error, which was computed as the area of the space obtained by applying an exclusive OR operation to the two areas of the target and user-drawn shapes (so that if the shape was perfectly traced, the two areas overlapped completely and therefore the error was 0). This method of shape error calculation is very straightforward and does not have to deal with corresponding vertices between shapes, as with Zabramski's technique [212]. The contact point error modelled the accuracy of initial targeting of the shape by the user, whereas the shape error represented overall tracing precision for the entire stroke. The hypothesis here was that the pointer-down error would be higher for finger tracing, but that the shape errors might not turn out to be significantly different, since users could automatically adapt their tracing movement as they visualised where the stroke appeared.

4.4.2 Results

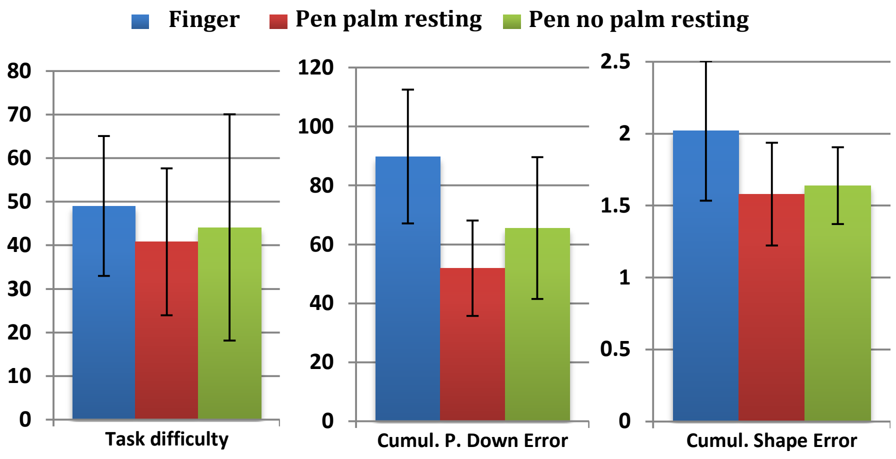


Figure 4.5: User ratings of task difficulty (from 0=extremely easy to 100=extremely difficult), cumulative pointer-down error (in px) and cumulative relative shape error

Figure 4.5 shows users' difficulty ratings and the two errors described above. As can be seen, this task received a range of assessments regarding its difficulty, with average ratings for all three configurations between 40 and 50 and evidently no significant differences. In the discussion session after the task, participants who judged the experiment relatively difficult gave the time pressure imposed on them for some of the shapes as the reason for their ratings. This consideration overshadowed the differences between the three input methods and so users were asked directly how comfortable they had been with pen or finger tracing and if they would have preferred an alternative technique. While 5 participants of touch tracing said they would rather have used a pen, there were also people in the pen conditions (2 in each group), who expressed a preference for direct touch input, which was somewhat surprising. Regarding palm resting, 4 participants of the third group explicitly stated they would have liked to have had that possibility. As for precision, the error charts of Figure 4.5 seem to confirm the hypothesis that only the pointer-down errors would exhibit considerable disparities between touch and the two pen configurations. At first glance, it does not appear as if palm resting had any significant influence. But breaking down the results of the tracing subtasks according to their associated time settings presents a different picture. Figure 4.6 shows the separated error values.

ANOVAs were conducted on the 6 dependent measures to identify significant differences. The numeric results of those analyses and of the post-hoc pairwise comparisons (when relevant) are reported in Table 4-1, with significant values highlighted in bold and values close to the α-level of 0.05 (specifically before and after Bonferroni correction) in italics.

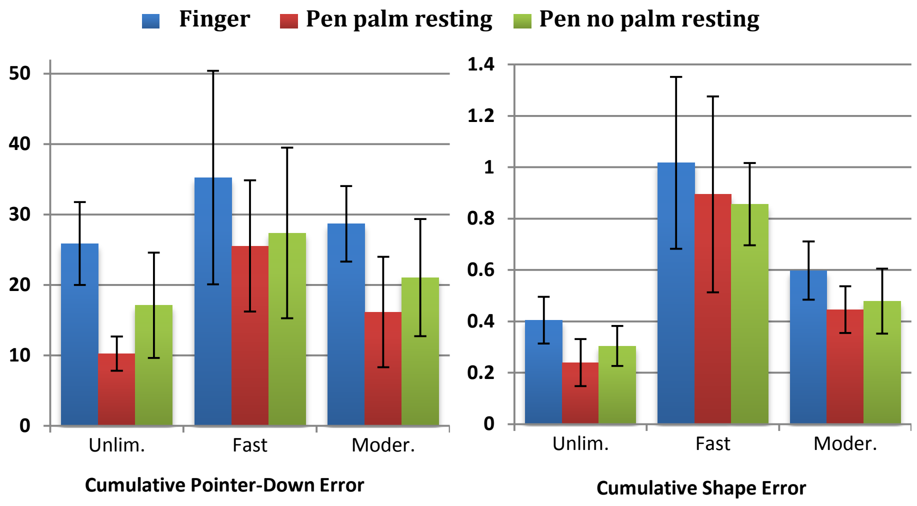


Figure 4.6: Cumulative pointer-down error (in px) and cumulative relative shape error for the three time constraint settings unlimited, fast and moderate

The data reveals interesting results, most strikingly that there is no significant effect of palm resting on overall tracing precision. Even the pointer-down error in the unlimited time setting is not significant enough (after Bonferroni adjustment) to draw clear-cut conclusions. Furthermore, it appears that, in fast drawing contexts, finger or pen perform equally well (or rather equally badly), although the large standard deviations indicate that this varies considerably among users. The pen with palm-resting, however, is always more accurate than touch input in conditions with no or moderate time pressure, which is consistent with expectations and Zabramski's results [212].

	Pointer Down Error	Shape Error
Unlimited	F=19.08, **p<0.001**	F=9.17, **p=0.001**
Fast	F=1.72, p=0.199	F=0.75, p=0.482
Moderate	F=7.48, **p=0.003**	F=5.13, **p=0.013**

ANOVA

	Pointer Down Error		Shape Error	
	Unlimited	Moderate	Unlimited	Moderate
Finger vs. Palm resting	**<0.001**	**=0.002**	**=0.001**	**0.014**
Palm resting vs. no palm resting	*=0.030*	=0.307	=0.233	=0.790
Finger vs no palm resting	**=0.005**	=0.068	*=0.040*	=0.061

Post-hoc Tukey tests

Table 4-1: F and p-values for ANOVAs and p-values of post-hoc Tukey tests (when applicable) for the pointer-down and shape errors in the three time-constraint settings

Attempting an interpretation of these results and what they could imply for pen vs. touch interfaces involving rapid tracing, one can allege that opting for one or other input method has no or little consequence on accuracy (a consideration that arguably is not all that important in those types of scenario anyway). In more relaxed situations, such as artistic drawing and painting, the pen is clearly superior, even more so if those activities involve executing several short strokes rather than long freeform paths as in this task (i.e. the pointer-down error becomes a much more important factor). As for palm-resting, the influence on precision is less pronounced than initially thought, at least on a digital platform with all its imperfections.

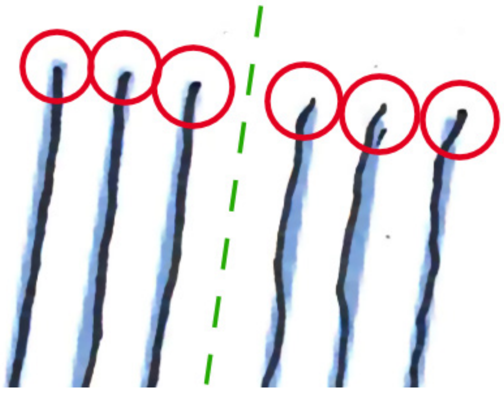


Figure 4.7: Lines traced on paper with arm rested on the drawing surface (left) and lifted (right). Starting portions of the line are circled in red.

After establishing those results, an informal experiment with a similar task using real pen and paper was performed. 6 people were asked to trace over a set of shapes and short lines printed on two sheets (without time constraints), where they had to lift their arm while drawing for one of the sets. The results more or less mirrored the experience on the tabletop. The shapes were drawn relatively accurately overall in both conditions but the short lines less so (Figure 4.7). It seems therefore that the most significant precision discrepancy between the two conditions is the initial targeting and placing of the pen. An initial firm stand on the surface allows more precise pointing than a shaky hand making an unsteady approach. But once contact has occurred, the necessary support is established and the tracing movement can continue with relative stability, albeit with slight fluctuations and irregularities.

Of course, it has to be added that the ability to rest the hand or the arm on the surface while executing drawing tasks is very important for comfort and fatigue. Protracted pen use with a lifted limb causes exhaustion and pain and so this problem needs to be addressed if only for that reason.

To summarise the findings in this experiments:

1. *Pen and touch have comparable, low precision in rapid tracing tasks.* The pen is more precise with no or less temporal demand.
2. *Palm resting mainly affects initial targeting and pointing accuracy of the pen. Overall tracing precision is also diminished, but to a lesser extent.*
3. A majority of users agree that the pen is more adequate for a tracing task. *There are, however, a few people who prefer direct touch input.*

4.5 Experiment 3

4.5.1 Description

Motivation

As has been pointed out previously, when the pen is not monofunctional, i.e. it does not only ink, the problem of how to change its mode rapidly and with minimal cognitive overhead arises. The main strategies to do so have been presented before and they involve either the pen itself with explicit or implicit switching or the NDH with user- or system-maintained activation. In an editing environment, it is likely that one mode, i.e. inking, will be used more, with other modes only sporadically triggered for occasional interleaved command actions. In such a context, therefore, users have to explicitly initiate the mode change, perform the command and revert to inking mode upon completion of the action(s). The cost to be in the default mode should be minimal, but at the same time, the interface should be designed so as to avoid switching errors. The most common way of enabling a function on a touch interface is probably by tapping a button of a toolbar, with the finger or the pen, where ideally the interface allows either input type to be used. Another popular alternative that can be seen in recent UIs is the in-place pop-up menu (in touch interfaces, often of the radial kind), summoned by a tap or a long press on the screen and selections within it are made by a second tap on the desired option or by sliding the finger to the corresponding trigger area (OneNote, for instance, uses this method).

As explained in Chapter 2, there is evidence in previous work that in the case of bimanual interfaces, quasimodes [160], i.e. user-maintained modes through continuous muscular tension can bring benefits to mode switching, in particular in a setting, where a default pen mode is changed at intervals only. One system that explicitly takes advantage of NDH-induced pen-mode switching is SpringBoard [89]. The technique relies on a physical actuator to trigger menus from which modes are selected by the pen. The actuator has to be maintained pressed to keep modes active and releasing it reverts to the default mode. The study conducted by the authors show that users prefer SpringBoard to standard selection techniques. Because their system uses a single physical button, mode-switching is a two-step process, in which the actuator first signals a change and the DH then has to indicate the specific mode to enter. The availability of simultaneous pen and touch allows more straightforward designs, in which the quasimodal action of the NDH directly sets the mode. A question that one might ask, for example, is if a regular toolbar can be used in a quasimodal manner, i.e. with users having to keep a finger down on target icons to trigger and maintain particular modes. For regular modal activation, where both pen and touch selections are allowed, it is furthermore interesting to investigate if people rather make use of their NDH to operate the toolbar (assuming it is located in its vicinity) or use the pen to perform all actions. Prior work so far suggests that there is no dominating preference among users [75, 93].

Design

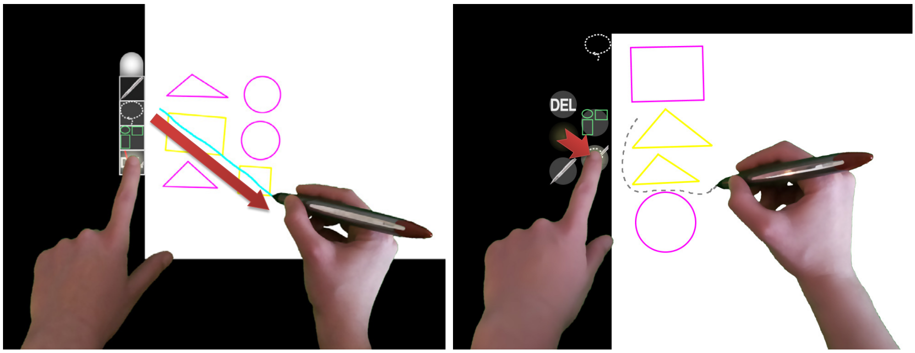


Figure 4.8: The mode-switching task. Delete function with the toolbar (left) and lasso selection with the pop-up radial menu (right)

With the above goals and context in mind, an experiment was designed around a mock drawing program, in which users had to sketch different geometrical shapes and occasionally execute non-inking operations on them. This approach differs from the one used in the SpringBoard study, which consisted in sets of controlled mini-tasks with a single repetitive marking action (circling a dot inside a box). Here, a more open-ended design was favoured to give more freedom to users and make them feel more as if they were using a real application. This choice precludes direct comparisons between single interactions and motion paths but on the other hand, it allows more spontaneous user behaviour, the analysis of which was more the intent of the study. Hence, a protocol was created, where the main task consisted of drawing shapes on a worksheet with occasional mock non-inking operations to be performed on those shapes. Specifically, participants were asked to trace a series of freehand triangles, rectangles and circles, with no constraints as to where and how the shapes had to be drawn (apart from having to be within the bounds of the worksheet). The shapes were then recognised by a shape recogniser engine and the strokes vectorised accordingly on the worksheet. This step was introduced for two reasons. First, it was to make sure that shapes were properly categorised by the system when manipulations needed to be performed on a specific subset of them. Second, the use of recognisers made for a better test environment in anticipation of editing applications using in situ and on-the-fly conversion of user ink input.

To minimise training time for study participants, the non-inking operations were limited to just three functions: lasso selection, selection by example (i.e. a particular query shape had to be drawn to select other similar shapes) and deleting. The techniques used to trigger those functions formed the three configurations of the experiment: toolbar simple tap, toolbar maintained tap and pop-up radial menu. In the first configuration, pen modes, including inking, were simply selected by tapping corresponding buttons in the toolbar once. In the interest of putting the "touch manipulates/pen draws" division of labour principle to test, users were given the possibility to tap buttons either with the pen or with fingers. For the second configuration, the default inking mode was enforced, that is, when the toolbar was not touched, inking mode was automatically selected. To activate a non-inking function, users were required to tap with and maintain a finger of their NDH on the corresponding button during the execution of the command (Figure 4.8 left). Releasing the finger caused the mode to immediately revert to inking. For both toolbar configurations, the widget was floating, i.e. it could be freely moved on the surface and placed at a convenient location. At the start of the experiment, the toolbar was placed on the side of the worksheet corresponding to the user's NDH (i.e. left for right-handed users and vice-versa). The third configuration featured a pop-up radial menu called with a single tap outside the worksheet and a long press when inside it (in order to differentiate the gesture from panning interactions). A function could then be chosen by sliding the finger to the appropriate button and releasing it (Figure 4.8 right). An icon showing the currently selected mode was displayed next to the worksheet for user awareness.

The task comprised 30 operations: 21 tracing operations and 9 commands evenly divided among the 3 available functions. The instructions about the operations to perform at each step were given via pre-recorded audio to ensure that users could remain visually focused on the interface. Examples of instructions given to users were "draw a circle", "lasso the rectangles", "delete the triangles" etc. For non-inking commands, other than specifically naming the type of target shapes to be edited, the latter were also highlighted for increased clarity and awareness. As soon as an operation was successfully completed the next audio instruction was given, until the end of task. If the wrong tool was selected for an operation this was signalled to the user by an appropriate audio message (and the error was recorded by the system). For delete and lasso operations, users were able to strike/select all target shapes in one stroke or in succession using multiple strokes.

4.5.2 Results

Generally, participants did not experience any major difficulty for this experiment. This is reflected in the scores as well as in the qualitative feedback, which was overwhelmingly positive. Comparing the three configurations with each other, Toolbar Maintain seems to have been rated slightly less easy, but an ANOVA revealed no significant differences between the means ($F_{2,27} = 0.737$, $p = 0.488$). No main effects were observed for completion time either ($F_{2,27} = 2.614$, $p = 0.092$). Regarding pen-mode switching, all users of the three groups were comfortable with their tool-selection technique and very few errors were made. In fact, the majority of participants completed the task flawlessly. The average number of errors made by participants was below 1 for all configurations, suggesting that changing the function of the pen from time to time is not a problem (at least, when the task is simple enough). The low number of mistakes contrasts with the higher error rates obtained in the SpringBoard study, but that is not surprising considering the latter's subtasks were much shorter and alternating was more frequent. However, even though their mode-triggering menu designs were different, the authors also did not observe main effects on error rate and completion time.

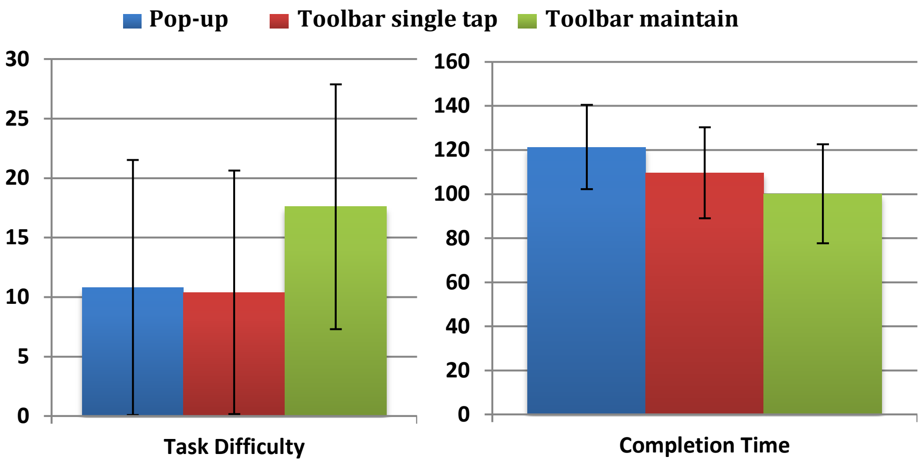


Figure 4.9: Task completion times (in sec.) and user ratings of task difficulty (from 0=extremely easy to 100=extremely difficult).

A somewhat surprising result obtained with the Toolbar Single Tap group was the fact that only half of the users selected commands with touch. The other half opted for the pen (with one participant using both). As stated in the description of the study protocol, the toolbar was located near the user's NDH and so to minimise movements for command activation and pen switching, touch would have been the optimal choice. Nevertheless, many people used the pen for all operations of the task, even though it involved unnecessary back-and-forths between the worksheet and the toolbar. A possible explanation for this behaviour is that users wanted to remain focused on a single interaction medium instead of dividing their attention between two separate areas of the interface. Moving the pen towards the toolbar in synchronisation with their gaze allowed smooth and uniform transitions, compared to context switches performed perhaps more abruptly when only the eyes shifted from one hand to another. Another possible reason is that people are used to interact unimanually with their devices and so their habits naturally transfer to other environments. Anyhow, this result further supports the view that the division of labour should not be strictly enforced and that users should be given the choice to use touch or pen, when possible.

In terms of number of tool selections, the Toolbar Maintain configuration was expected to show the lower counts and indeed this turned out to be the case. Compared to other configurations averaging over 17 explicit command activations, participants of the Toolbar Maintain group managed with only 11 taps on average. Interestingly, however, measures of how long users were in contact with the surface revealed that the average times for Toolbar Maintain and Pop-up Menu were comparable (25.5 sec and 25.4 sec average times respectively), which suggests that the amounts of interaction effort of the NDH in the two techniques are more or less similar (excepting the sliding motion for the Toolbar configuration), even though Pop-up does not require postures to be maintained. Of course, one cannot generalise that observation to all kinds of contexts, as finger-dwelling time for maintained activation of commands obviously depends on the time needed to execute them.

In general, it does not seem like there is any significant difference between traditional menu designs for pen mode-changing both in terms of user preference and performance. The use of quasimodes in those types of widgets does not appear to have any substantial effect either. But those observations cannot necessarily be generalised to all types of tasks with different levels of complexity and mode-switching frequency, as shown by the different results and error rates obtained in the SpringBoard study.

To sum up:

1. *In tasks where inking is the dominant mode for the pen, switching using pop-up menus, toolbars with and without maintained postures are roughly equivalent in terms of efficiency and user acceptance.*
2. Users do not necessarily perceive the pen to be exclusively a tracing or precision instrument, as it is also used for manipulations also performed by touch to a non-negligible extent.
3. *Quasimodes can slightly decrease the interaction effort if non-inking pen commands are used sporadically.*

4.6 Experiment 4

4.6.1 Description

Motivation

The previous experiment considered mode-switching by means of common UI widgets such as toolbars and in-place pop-up menus. It also considered quasimodal activation but the results did not conclude to any particular advantage. Prior work, however, suggests that in some circumstances, quasimodes can increase awareness of the currently active state and thus reduce errors when changing it [89, 173]. Those studies considered a single actuator to trigger mode changes, which minimised the cognitive effort required to perform this support action. Specifically, by virtue of being a simple on-off trigger in a fixed location, the actuator enabled users to perform "blind" mode modifications without having to look at the limb executing the activation action. As discussed in the motivation of the previous experiment, however, a single actuator either limits the mode change to one alternate state only, or requires a multistep activation (as with SpringBoard). It is not clear if increasing the number of actuators would be able to support direct mode selection without losing the benefit of the low cognitive load on users, as additional action and hence concentration would be required to move to the desired button or pedal and then trigger it. To be able to do that in a blind automatic way would likely necessitate training of spatial memory, which can be a burden if the locations of multiple mode actuators need to be learned.

As introduced in Chapter 2, an alternative method to trigger mode changes with the NDH is through postures such as chords or particular contact shapes on the surface. Such postures allow direct activation of multiple modes and, if not constrained to particular locations of the user interface (as the command pad of [85] for example), seem to fulfil the requirements for blind switching, as users need not care about *where* their hand is touching but *how*. Furthemore, if the activation of pen functions through the touching hand is static, i.e. the switch to a different pen mode is never triggered by a gesture requiring a complex motion of the NDH (in accordance with Guiard’s kinematic chain model), the cognitive demand remains low.

While unconstrained posture-based mode-changing has been utilised in some research prototypes (see again Chapter 2), those techniques have never really been studied in the context of bimanual coordination. It is worth finding out whether they are indeed able to exhibit the benefits of quasimodes and if that is the case, to what extent.

Design

To test people’s ability to switch modes effectively, while maintain focus on the main task of the dominant hand, the following experiment was devised: participants were asked to trace over a polyline, whose segments had different line styles, in a single stroke. Users had to change the style on-the-fly using the NDH to correspond to that of the segment they were currently drawing with the pen-holding DH, without lifting it. This task has some similarities with tape drawing (and its digital counterpart [33]), which is a technique to create smooth curves using asymmetric contributions of the two hands (the DH traces while the NDH sets the curvature of the curve). The differences here are that the changes triggered by the NDH are discrete and the focus changes from one hand to the other instead of the two moving continuously together.

Since the objective was to test coordination and mode-changing errors rather than accuracy, the goal for participants was less to trace over the polyline as precisely as possible, as in Experiment 2, than to execute the stroke quickly enough while correctly matching the styles of the target segments. For the same reason, time limitations were not included.

Compared to the previous experiment, this task did not attempt to directly mimic a real interface (there are indeed relatively few practical situations in which a pen's function needs to be changed in action). The purpose of its design was to force rapid synchronisation of the two hands in order to study the efficacy of the mode-switching techniques.

Again, as with previous tasks, the interface consisted of a worksheet (static, as in Experiment 2, in order not to corrupt error measurements) in the centre of the display on which the polylines appeared and tracing was to be performed. To keep the experiment tractable and easy to learn in a few minutes, only four different stroke styles were used: normal, dotted, dashed and bold. The three configurations that were set up controlled the way the NDH could activate the style changes. In the first two configurations, styles had to be activated using specific touch postures performed anywhere on the surface surrounding the worksheet (Figure 4.10 left). The difference between the two settings was that, in the first case, postures had to be maintained as in the previous experiment with the toolbar, i.e. fingers had to be held down to keep a non-normal stroke style selected (the normal stroke was activated by default when there was no touch contact), whereas, for the second configuration, a simple tap on the surface was sufficient to change styles (including normal style).

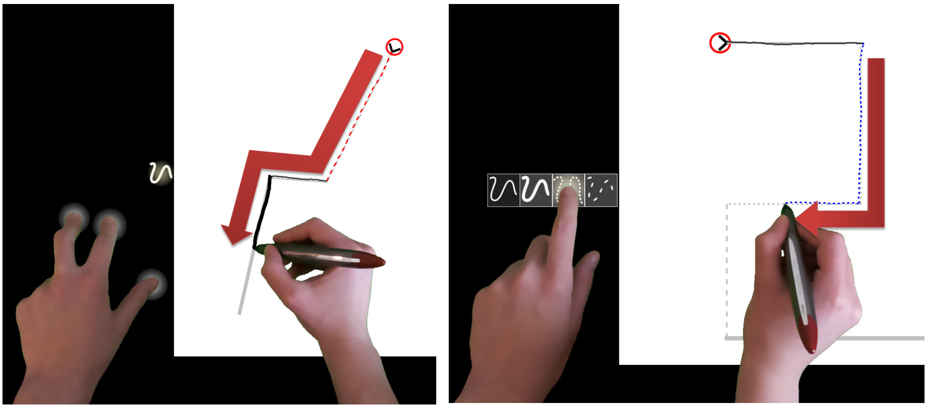


Figure 4.10: Bimanual coordination task with style changes controlled by postures (left) and toolbar selection (right)

The following mappings were defined for the four different styles: one finger down for dotted, two fingers for dashed, three fingers for bold and a gentle palm press for normal style in the second tap-to-select configuration. As a simple awareness mechanism to indicate the currently selected stroke style, an icon placed close to the worksheet was used. For the third baseline configuration, a classic toolbar, similar to the one used in the previous experiment (but horizontal) was displayed with single-tap buttons corresponding to each style (Figure 4.10 right). In all configurations, palm resting on the worksheet was allowed.

Other than the measurements common to all tasks, three important pieces of data were recorded: the total line-tracing error, the number of stroke style errors and the pen dwell time. The first quantity represented the total tracing error when attempting to draw the polyline regardless of style. It was computed by simplifying the user-traced curve into a fixed number of points and calculating the distance between those points to the closest segment on the target polyline (this method is similar to Zabramski's [212]). The second value, the stroke style error, measured how many mismatches occurred between the style of the current portion of the user curve and that of the corresponding segment of the target polyline. Specifically, an error was registered each time the distance between the current styled point on the user curve and the nearest point with the same style on the polyline exceeded a particular threshold value (this distance was infinite if the currently selected style was not present in any of the segments of the polyline). This was only done for distinct pairs of mismatched styles, i.e. no new error was logged if the next point on the user curve had the same mismatched style as the previous one.

The third quantity that was kept account of was the pen dwell time, which was defined as the amount of time the pen did not move (or moved very little) during the tracing movement. Those pauses mostly occurred when the user stopped the tracing motion to change styles between segments. This time could then be compared to the total stroking time of the polyline, given the number of required style changes to trace it. A high dwell time for a stroke would indicate longer periods of style change activations and more synchronisation effort. Higher dwell times were expected for the toolbar configuration, as users had to momentarily shift their gaze and concentration from the tracing hand to the operating hand that had to select the correct style button on the toolbar (as in the previous experiment). In the posture configurations, however, people would likely remain focused on the tracing activity and trigger the different styles without looking, by using the appropriate finger/palm combinations, almost as if playing the piano. If users were able to master that technique, it would hopefully lead to shorter pauses between segments, although perhaps also with the consequence that errors would be more frequent due to the rapid and un-checked switching activity.

4.6.2 Results

Here also completion time and task difficulty results exhibited a similar pattern, with the first "Posture maintain" configuration seeming to be both the easiest and the fastest (Figure 4.11). ANOVAs revealed, however, that only the former had statistically significant differences, using 0.05 as the cut-off α-value ($F_{2,27} = 4.06$, $p = 0.029$ vs. $F_{2,27} = 2.76$, $p = 0.082$). Tukey tests showed that the lowest p-value between "Posture Maintain" and the toolbar configuration was 0.025, which is higher than the threshold value after Bonferroni adjustment (0.017) but only marginally. One can therefore only discern a trend, but no clear general distinctions between the three configurations.

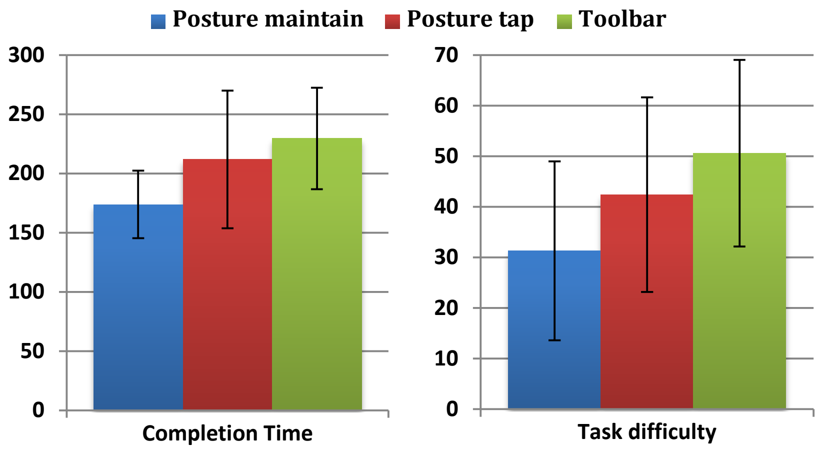


Figure 4.11: Task completion times (in sec.) and user ratings of task difficulty (from 0=extremely easy to 100=extremely difficult)

Feedback from the participants was also very diverse. For the posture configurations, people appreciated the ability to concentrate on the pen-holding hand actually performing the tracing motion, although a few users felt the need to constantly check whether the correct tool had been selected. 3 people critiqued some of the mapping choices between postures and styles (e.g. the palm posture in "Posture tap" to revert to the normal style) and 2 suggested that the visual clues to maintain awareness of the currently selected style should be improved. For the toolbar configuration, there were fewer comments relating to the input method, which is understandable considering its commonness in user interfaces.

Although some disparities were originally expected between older participants less familiar with touch interfaces and power users, it turned out rather surprisingly that the former did not fare significantly worse than the latter. People in the Posture Maintain group were able to master the bimanual pen mode-switching technique, regardless of their initial skills. This is an encouraging result for the development of future interfaces based on this interaction method.

The detailed tracing times and errors, reveal more telling disparities (Figure 4.12). Statistical tests for the three metrics confirmed main effects for the stroke style error ($F_{2,27} = 6.74$, $p = 0.045$) and pen dwell times ($F_{2,27} = 15.75$, $p < 0.001$) (for the tracing error $F_{2,27} = 2.64$, $p = 0.09$), with post-hoc comparisons yielding significant mean differences between "Posture Maintain" and "Toolbar" for the style error ($p = 0.003$) and between "Posture Maintain" and both other configurations for dwell time ($p = 0.002$ and $p < 0001$ respectively).

Thus, the data shows that while users were mainly quicker with the posture configurations they were also more careless. Interestingly, forcing the selecting posture to be maintained in the first configuration seems to have encouraged that behaviour even more, compared to the second configuration that required only simple tapping to change the style. This is consistent with observations of users performing the task, which showed that, in the first case, participants checked less often that their finger postures triggered the right style selections, compared to the other two tapping configurations. Those shorter coordination times for "Posture Maintain" are reflected in the pen dwell times, which average only 20% of the total time the pen was in contact with the surface, compared to roughly 33% for the other configurations. The difference between those proportions is possibly also partly due to the fact that a retained posture sometimes only required lowering or lifting fingers (e.g. to switch from dotted to dashed style), whereas a full tapping gesture necessitated a movement of the whole hand with correct placement of the fingers.

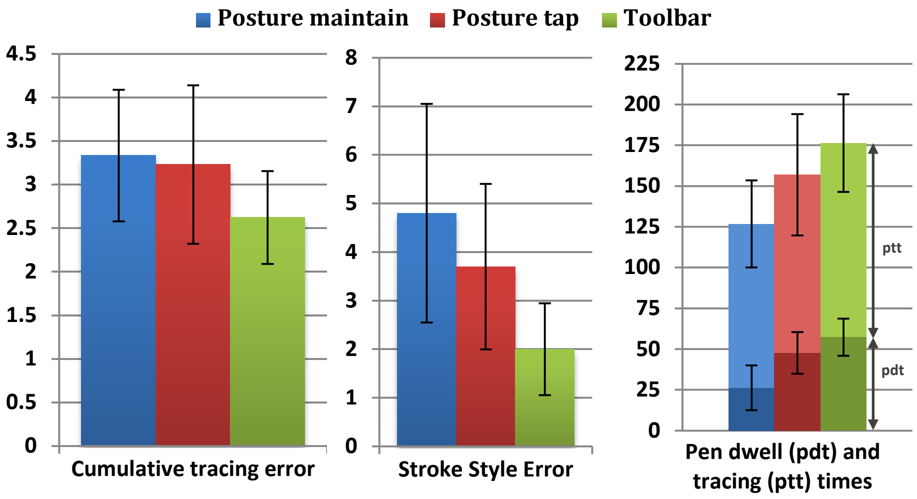


Figure 4.12: Cumulative relative tracing error, number of stroke style errors and pen dwell and tracing times

The following can therefore be learned from those results:

1. *Pen mode-switching or function activation through postures can speed up bimanual coordination and synchronisation but at the cost of increased errors. This may mainly be a question of practice, however, as postures can prove effective as functional shortcuts in bimanual pen and touch applications once the necessary finger and hand positions have been acquired. To some extent, this process can be likened to learning how to coordinate the hands when playing the piano.*
2. *Maintained postures encourage further speed increases* and require less coordination effort.

3. Adequate awareness mechanisms are needed when employing blind NDH interaction techniques so that users can make sure their actions produced the desired response from the UI. *Because the user's focus is on the DH and not the NDH, such feedback should probably either be close to or on the pen (it could, for example, change colour to reflect the active mode) or be subtly integrated in the interface so as to gently inform the user, without disrupting the execution of their task.*

4.7 Conclusion

The four experiments conducted in this study corroborated some results reported in prior work and yielded new findings. In the former category is the confirmation that allowing the interchangeable use of pen or touch for a particular type of manipulations (e.g. menu selections, button activations) is a sensible choice. Interface designers must however be careful to set clear delimiters between those widgets and the inking space, when overloading touch controls with pen interactions. To avoid errors, it should always be clear to the user whether the pen will perform its default marking function or if it will cause a different response. Needless to say that UI controls should be appropriately sized to be effortlessly operated both by pen and touch. Of course, there are cases where touch is the only obvious possible input method, for example for operations involving multipoint gestures (zooming, rotating etc.) and interactive spaces where touch and pen cause different actions to take place (e.g. on a worksheet: touch pans while pen inks). But for clearly monofunctional areas such as toolbars, buttons, and menus, limiting activation to touch seems like an arbitrary and overly restrictive design choice. With regard to performing unaided multiple transformation operations with a single action, the first experiment demonstrated that, when a certain level of accuracy is required, the recommendations to separate degrees of freedom in 3D also apply in 2D. Precise manipulations are best broken down into independent atomic actions, on which users are able to fully concentrate. As for pen-mode changing, the last two experiments provided further evidence that the NDH can be an effective activator. The fourth task additionally showed that spatially-unconstrained chord-postures are especially powerful, if properly mastered, as they allow users to focus their attention on the work performed by the DH. Moreover, because this technique builds on natural automatisms of bimanual motor skills, this type of technique is not the privilege of particularly dexterous people. Still, as with most controls and selectors, appropriate UI feedback is required to maintain user awareness.

Another important contribution is the realisation that palm resting affects initial targeting with the pen to a greater extent than stroke precision, once the pen is touching the surface. This result suggests that solving the palm-resting problem is less critical for applications requiring infrequent use of the pen (because of fatigue) and where that use is limited to coarse actions or long strokes, e.g. for selections or brief inking. Those conditions clearly exclude activities such as artistic sketching and painting, which typically involve successions of minute strokes. The latter cases call for appropriate measures to filter out incidental touches from resting arms in order to guarantee a satisfying user experience. Yet no matter how robust the technique may be, it will most likely never achieve 100% accuracy if highly flexible and precise differentiation is desired (i.e. no blanket discarding of one modality when the other is active). In particular, for pen and touch situations in which both input types are used in close temporal and spatial proximity (e.g. for the pinch hold + pen drag gesture), one must be careful with restrictive palm-rejection schemes that may compromise the efficacy of such gestures. Thus, the decision to include such filtering mechanisms depends on the application scenario as well as the pen and touch interactions to integrate.

5 Document Scenario 1: Active Reading

Active reading (AR) is the act of interactively perusing a document by employing effective strategies to better comprehend and remember its content. It is an important activity that is regularly performed in several contexts, including as part of office work. With paper documents and books, AR usually involves annotating, underlining or highlighting elements with pens or markers and searching through those documents for specific information [145]. With technological advances and the need to digitise information for more efficient integration and recollection, researchers and engineers have attempted to support those processes in various ways, taking care, when possible, to preserve the naturalness and intuitiveness with which people engage with physical documents by more or less mimicking the properties of paper interaction. Thus, devices of choice for prototypical AR solutions have mainly been pen-operated tablets [73, 168] and recently also multitouch tabletops [188]. Within that context, the move to pen and touch seems like a natural step to further explore how this task can best be supported with digital media, which is the subject of this chapter.

The outline is as follows: first, the system and its AR-supporting features (navigation functions, support for annotations, in-document searching and retrieval of external reference material) are introduced. Then, the results of an evaluation aimed at showing to what extent the application is able to effectively combine the advantages of paper and that of a typical digital platform such as Adobe Acrobat on a PC, but without their respective disadvantages, are reported. Qualitative data about what types of gestures are most effective for such document tasks is also provided. Finally, from those experiments and user feedback, important hardware and software issues that particularly stand out in such kinds of document tasks are identified and from that analysis, a number of system requirements are derived.

Part of the material of this chapter was previously published in the proceedings of the AVI 2012 conference [142].

5.1 Related Work on Active Reading

In their seminal 1997 study, O'Hara and Sellen compare people's active reading experience on paper vs. on a computer screen [148]. They observe that affordances of paper are compelling and derive design implications for digital systems to better support the task, in particular the need to provide adequate tools for annotations, quick navigation and control over spatial layout of content. 10 years after, Morris et al. conduct a study based on similar experimental conditions, in which they highlight the progress made by computers [145]. They report, for instance, that tablets are able to perform on par with paper in their AR tasks.

A more recent investigation by Tashman and Edwards reinforces the impression of computers' growing dominance over paper for AR [187]. The authors also notice that as digital tools have become more pervasive, people's habits have also changed so that AR tasks are now more readily executed on a digital device rather than on paper. To demonstrate that AR can be effectively done on multitouch surfaces, the researchers develop LiquidText [188]. Contrary to most of the previous systems designed for that activity, LiquidText deliberately breaks with the paper/page metaphor for the document layout to give users more flexibility when viewing different sections of a document. This is achieved through collapsing actions with pinching gestures causing document areas to be virtually folded together. Interestingly, the authors also reveal that they would have opted for a pen and touch interface had they managed to find adequate hardware.

Finally, Hinckley et al.'s GatherReader is an e-reader application running on a pen and touch-enabled tablet, which especially focuses on the information gathering aspect of active reading [88] (see also summary in the "Pen and Touch Applications" subsection of Chapter 2). The authors compare their content extraction technique to that of LiquidText by highlighting a compromise between precision and transaction cost to select and copy text sections. Their *Frame-and-Drag-to-Pocket* method sacrifices some precision by allowing only document sections spanning the whole page width to be framed, but at the benefit of a lighter gestural phrase [54] to then copy the desired section. In the interest of keeping the focus on the document so as not to disrupt the main task of reading, the authors integrate a split-screen view for freeform note-taking with a notebook page appearing side-by-side with a page of the actual document. The document and the notebook can then be flipped independently with single-page views only. This design has the advantage of maintaining the locus of action within a small area, but it also breaks the normal double-page view of the document, which may feel somewhat disconcerting (say if the user wants to annotate an illustration spanning a double page). What is more, the relaxed full-width selection method applied to notes attached to the original content does not seem to allow the linking of side-by-side annotations to different pages of the document. There is also no possibility to see all document pages at once with their associated annotations in a thumbnail or similar overview screen.

5.2 The System

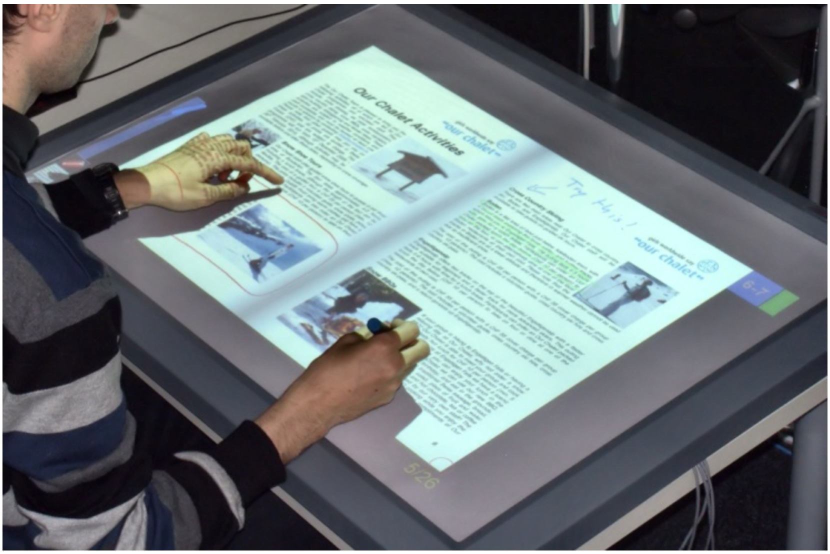


Figure 5.1: The system in action

5.2.1 Design Approach

Considering the target environment and input paradigm, which attempt to remain close to people's natural experience with physical documents, it seems sensible to opt for the path of the paper metaphor with adequate enhancements that a pen and touch platform permits (as the authors of GatheredReader have also recognised). This design approach is further justified by the fact that PDF documents, arguably the most common format for rich documents intended for reading, use fixed, page-based layouts and hence lend themselves well to a close-to-paper setting.

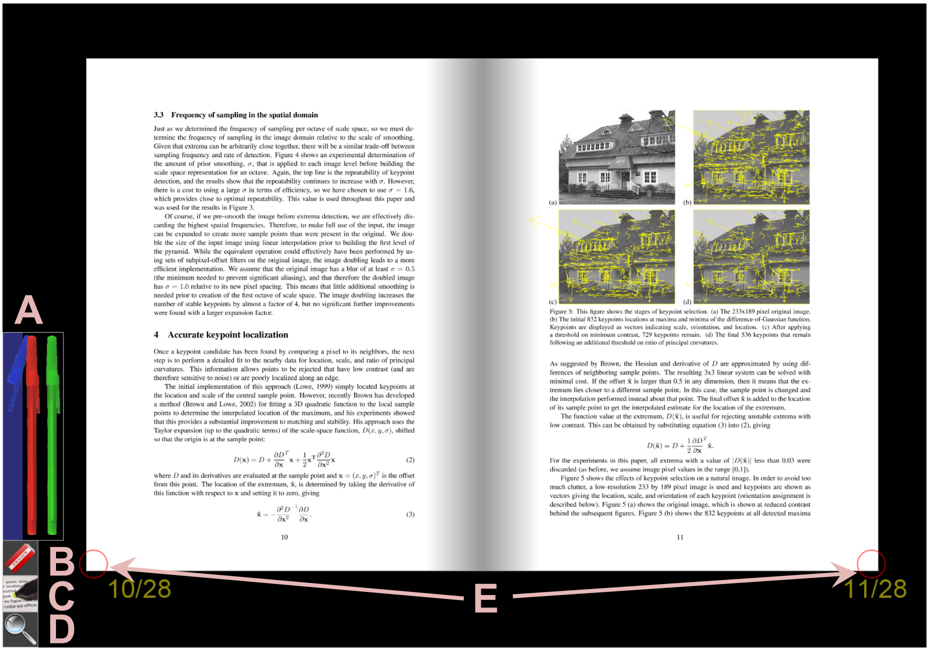
Given the main processes of AR outlined by O'Hara and Sellen, i.e. annotating, navigating and laying out content, the following requirements are defined for the application prototype:

- Annotating: adding, deleting and recalling of annotations
- Navigation: Sequential and random movement in the document. The user has to be able to rapidly browse through the pages of the document and easily locate relevant information.
- Layout: overview of the whole document to gain a quick panorama of its content at a glance.

With regard to navigation, perhaps one of the most useful features available for targeted browsing of electronic documents but lacking for their paper counterparts is keyword searching.

Such functionality can of course be easily provided through a virtual keyboard, but this would somewhat stray from the pen and paper metaphor and be incongruous within a pen-supported active reading environment, where users interact with content directly. Therefore, it was decided to include keyword-querying functionality entirely via the pen (with appropriate mode-switching), i.e. keywords are directly handwritten by the user or selected in the document, similar to techniques used in InkSeine [94].

5.2.2 User Interface



3.3 Frequency of sampling in the spatial domain

Just as we determined the frequency of sampling per octave of scale space, so we must determine the frequency of sampling in the image domain relative to the scale of smoothing. Given that extrema can be arbitrarily close together, there will be a similar trade-off between sampling frequency and rate of detection. Figure 4 shows an experimental determination of the amount of prior smoothing, σ, that is applied to each image level before building the scale space representation for an octave. Again, the top line is the repeatability of keypoint detection, and the results show that the repeatability continues to increase with σ. However, there is a cost to using a large σ in terms of efficiency, so we have chosen to use $\sigma = 1.6$, which provides close to optimal repeatability. This value is used throughout this paper and was used for the results in Figure 3.

Of course, if we pre-smooth the image before extrema detection, we are effectively discarding the highest spatial frequencies. Therefore, to make full use of the input, the image can be expanded to create more sample points than were present in the original. We double the size of the input image using linear interpolation prior to building the first level of the pyramid. While the equivalent operation could effectively have been performed by using sets of subpixel-offset filters on the original image, the image doubling leads to a more efficient implementation. We assume that the original image has a blur of at least $\sigma = 0.5$ (the minimum needed to prevent significant aliasing), and that therefore the doubled image has $\sigma = 1.0$ relative to its new pixel spacing. This means that little additional smoothing is needed prior to creation of the first octave of scale space. The image doubling increases the number of stable keypoints by almost a factor of 4, but no significant further improvements were found with a larger expansion factor.

4 Accurate keypoint localization

Once a keypoint candidate has been found by comparing a pixel to its neighbors, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

The initial implementation of this approach (Lowe, 1999) simply located keypoints at the location and scale of the central sample point. However, recently Brown has developed a method (Brown and Lowe, 2002) for fitting a 3D quadratic function to the local sample points to determine the interpolated location of the maximum, and his experiments showed that this provides a substantial improvement to matching and stability. His approach uses the Taylor expansion (up to the quadratic terms) of the scale-space function, $D(x, y, \sigma)$, shifted so that the origin is at the sample point:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2}$$

where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. The location of the extremum, $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to $\mathbf{x}$ and setting it to zero, giving

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}}. \tag{3}$$

10

Figure 5: This figure shows the stages of keypoint selection. (a) The 233x189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

As suggested by Brown, the Hessian and derivative of D are approximated by using differences of neighboring sample points. The resulting 3x3 linear system can be solved with minimal cost. If the offset $\hat{\mathbf{x}}$ is larger than 0.5 in any dimension, then it means that the extremum lies closer to a different sample point. In this case, the sample point is changed and the interpolation performed instead about that point. The final offset $\hat{\mathbf{x}}$ is added to the location of its sample point to get the interpolated estimate for the location of the extremum.

The function value at the extremum, $D(\hat{\mathbf{x}})$, is useful for rejecting unstable extrema with low contrast. This can be obtained by substituting equation (3) into (2), giving

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}.$$

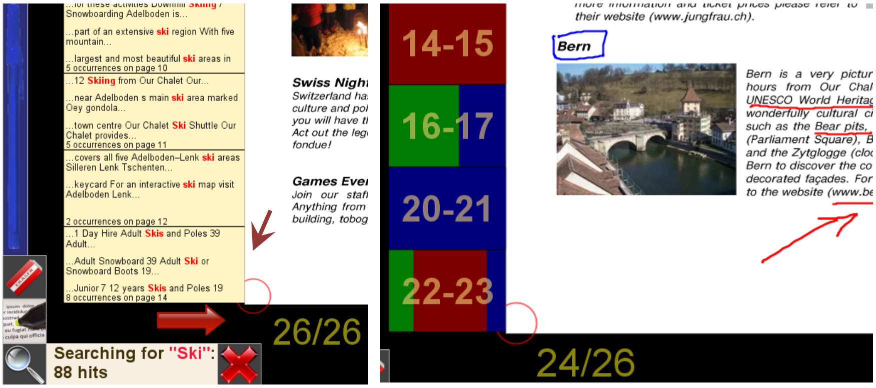
For the experiments in this paper, all extrema with a value of $|D(\hat{\mathbf{x}})|$ less than 0.03 were discarded (as before, we assume image pixel values in the range [0,1]).

Figure 5 shows the effects of keypoint selection on a natural image. In order to avoid too much clutter, a low-resolution 233 by 189 pixel image is used and keypoints are shown as vectors giving the location, scale, and orientation of each keypoint (orientation assignment is described below). Figure 5 (a) shows the original image, which is shown at reduced contrast behind the subsequent figures. Figure 5 (b) shows the 832 keypoints at all detected maxima

11

Figure 5.2: Main user interface and elements thereof for right-handed user (toolbar on the left)
A – Colour pen drawer
B – Eraser (delete) tool
C – Text-highlight function
D – Search-term input function
E – Interactive page corners (delimited by red rings)

The user interface is loosely inspired from 3Book [56], which adopts a 3D interactive codex book as its representational model, supported by a number of digital library features such as text search and highlighting, annotating and dynamic bookmarks. Those concepts developed for a PC application are partially borrowed and adapted for the pen and touch tabletop setting. Thus, the interface follows a virtual desktop paradigm, in which the working space of a physical desk is modelled and represented on the screen. Figure 5.2 shows the main user interface, with an open document and a command sidebar containing a virtual drawer with selectable colour pens (A) and the icons for the delete (B), text-highlighting and search-term input (C) functions. The interactive page corners (E), used to perform a number of page-turning actions, are defined by touch areas delimited by red rings, the diameter of which can be adapted to the user's preference. An overview screen, similar to Space-Filling Thumbnails [59], in which all pages of the documents are displayed as small icons (Figure 5.7), is available via a special gesture described further below. This overview mode breaks with the codex metaphor but it was identified as a useful addition, as there is often the need to view documents from a wider angle in order to quickly scan their content.


Snowboarding Adelboden is...

...part of an extensive ski region With five mountain...

...largest and most beautiful ski areas in
5 occurrences on page 10

...12 Skiing from Our Chalet Our...

...near Adelboden s main ski area marked Oey gondola...

...town centre Our Chalet Ski Shuttle Our Chalet provides...
5 occurrences on page 11

...covers all five Adelboden–Lenk ski areas Silleren Lenk Tschenten...

...keycard For an interactive ski map visit Adelboden Lenk...

2 occurrences on page 12

...1 Day Hire Adult Skis and Poles 39 Adult...

...Adult Snowboard 39 Adult Ski or Snowboard Boots 19...

...Junior 7 12 years Skis and Poles 19
8 occurrences on page 14

Searching for "Ski": 88 hits

their website (www.jungfrau.ch).

Bern is a very pictur
hours from Our Chal
UNESCO World Herita
wonderfully cultural c
such as the Bear pits,
(Parliament Square), B
and the Zytglogge (clo
Bern to discover the co
decorated façades. For
to the website (www.be

Figure 5.3: Text search tools. Dynamic page markers for search results with context snippets appearing on the side of the page (left) and bookmarks for annotated pages with stripes reflecting relative amount of strokes for each colour pen (right)

Another metaphor used in the application is that of sticky notes attached to pages as markers to be able to quickly return to particular sections of a document. The interface features virtual page markers, which appear dynamically at the outward side of pages, when hits from a search query located on non-visible pages need to be presented to the user (Figure 5.3 left). Those markers work as hyperlinks, which the user simply taps to immediately flip to referred pages. Annotations benefit from a similar treatment, in that edited content on pages currently not visible is referenced by corresponding bookmarks filled with multi-coloured stripes reflecting the proportion of markings penned using each annotation colour on said pages (Figure 5.3 right). This technique borrows from coding schemes used in collaborative authoring systems, where colours symbolise participating authors and editors and the amount of modifications made by each is shown in multi-coloured bars [150].

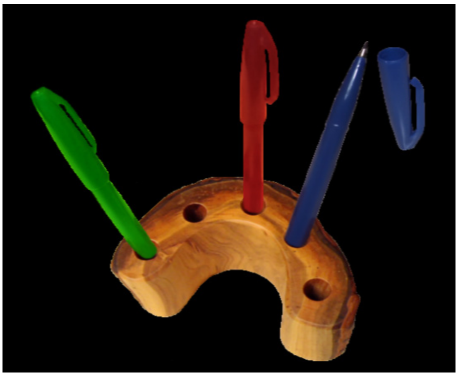


Figure 5.4: Initial design of the pen container

Figure 5.4 shows an initial design of the pen container as a virtual pen holder that the user can drag around the surface, in accordance with the virtual desk metaphor, but it was abandoned after realising that the object occupied too much space on the work area and often got in the way of the displayed document. The current sidebar is slim and compact. It docks on the side of the interface that is opposite to the user's handedness, since it is meant to be operated using the NDH.

This combination of different representational levels to present the user with the search or edit scope contributes to an enhanced awareness of the work context while remaining close to the familiar appearance of a physical document with inserted markers such as bookmarks and sticky notes. Furthermore, the ability to view snippets and symbolic representations of annotations at all time, regardless of the currently displayed page, contrasts with systems, such as GatheredReader, where users have to locate the page on which annotations were made to interact with them.

5.2.3 Gesture Set

The implemented gestures are a combination of classic unimodal multitouch interactions to manipulate digital objects (i.e. pan, zoom, rotate etc.) and a number of novel gestures exploiting the two modalities separately or simultaneously. As per the division of labour principle, the DH holding the pen is mostly used for inking, while the NDH performs coarser manipulations to position the document on the workspace and activate functions. However, it is also easy to perform bimanual multitouch actions by temporarily stowing the pen away inside the DH, as explained in Chapter 2. Hence all gestures are executable without burdening the user with time-consuming and effort-requiring context switches.

The following list presents some of the main gestures used in the application in two categories: "Multitouch" and "Touch + Pen", depending on the input types involved. The "Touch + Pen" gestures are executed by combining a simple single-finger touch action to change the mode and issuing the command with the pen. Function selection follows quasimodal mode-activation, i.e. a finger of the NDH needs to be maintained pressed on a specific interactive or hotspot area throughout the action carried out by the pen. Depending on the nature of the function, the execution of the corresponding command is then triggered by lifting either the finger or the stylus from the tabletop surface.

Multitouch

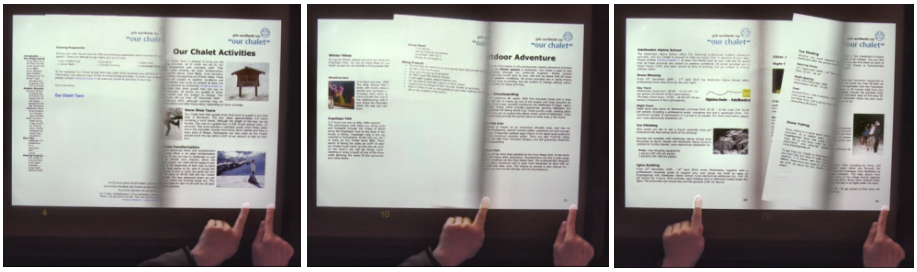


Figure 5.5: Forward multiple page flip: the right index activates the right corner and the left index moves laterally to flick through the pages

- Multiple page flip (Figure 5.5): In addition to turning pages individually by tapping their corners, users can execute this gesture to leaf through several pages of the document to rapidly browse through its content as if flipping the pages of a real book. It is initiated by pressing a finger on a page corner (typically the index finger) and sliding the fingertips of the other hand between the two lateral edges of the document to flick through its remaining pages. Depending on whether the left or right corner is activated, the pages before or after the currently opened page(s) are shown. The displayed page and the state of its folding animation are a function of the position of the moving finger(s) within the two lateral edges of the document, which represent the two end values for the page numbers (i.e. the currently opened pages when the gesture was initiated and the first or last page of the document). It is possible to perform several page-flicking operations consecutively by lifting the moving finger and running it across the document again, in a manner that is very similar to flipping the pages of a physical book.

- Document overview (Figure 5.6): the document overview is summoned by executing a chop-and-spread gesture where the two hands are placed vertically on the surface, palms facing, and then spread apart, as if scattering all pages of the document around. The document overview screen gradually fades in and replaces the main interface as the hands separate, until a threshold distance is reached and the overview completely fills the screen. Some level of relaxation is permitted in that a fingertip can be used for one of the hands instead of a fully stretched posture which may be more convenient for the pen-holding hand. This gesture is somewhat similar to Wu et al.'s Pile-n-Browse gesture [208] although it fulfils a different purpose. If the document contains a large number of pages, a multi-level overview appears, where only a subset of the pages are first displayed (e.g. pages with chapter titles or main sections if the metadata is available) and the user can view the remaining pages contained in a virtual "pile" by tapping its top page (Figure 5.7).

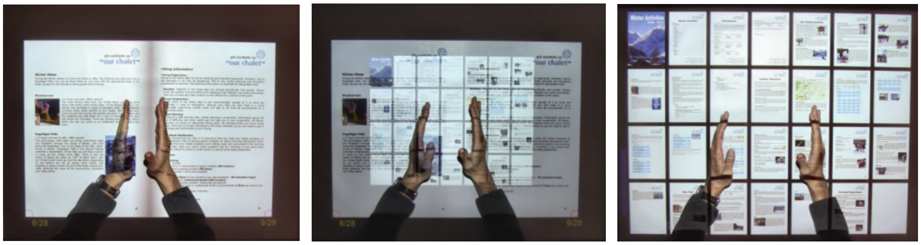


Figure 5.6: Chop and spread gesture to trigger overview screen

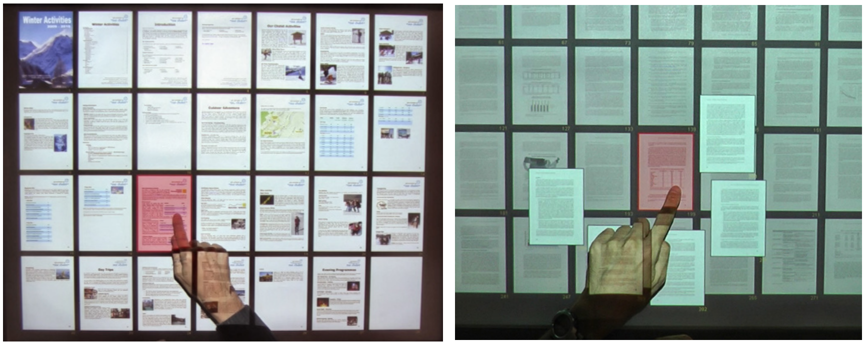


Figure 5.7: Direct page selection in simple overview (left) and spiral display of sub-pages in multi-level overview mode (right)

Touch + Pen

As mentioned previously, touch actions in this mode only involve pressing and maintaining a finger of the non-writing hand on an active area of the UI and is hereafter simply referred to as "activating".

– <u>Erase annotation</u> (Figure 5.8): annotations can be removed by activating the delete icon in the command sidebar and striking them out with the pen. Several annotations can be erased using a single stroke that intersects at least one of the lines that make up said annotations. Thanks to this combination that does not rely on error-prone gesture/stroke recognition or implicit mode-switching, annotations can be speedily and reliably erased.

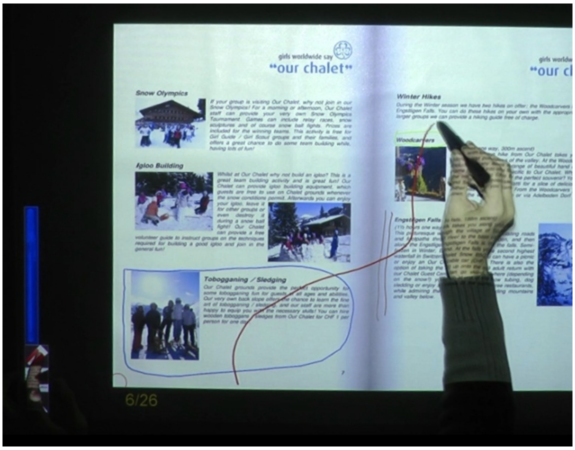


Figure 5.8: Deleting annotations with activated eraser function

– <u>Go-to-page</u> (Figure 5.9 left): it is possible to turn to a specific page of the document by activating one of the interactive page corners (the same ones used for flicking the pages) and writing the page number to turn to anywhere on the surface. When the finger is released, the handwritten number is recognised and the desired page is shown after a page-flipping animation. This feature is useful for non-hyperlinked document indexes and when users know to which page(s) they would like to navigate. In the index case, the user finds the page number of a chapter or section they wish to go to and simply writes that number to open the document to that page. An alternative would be to detect those references and recover the associated hyperlinks so that the user need only tap the link to navigate to the target page. But this requires document analysis, which might not always yield the expected result, especially if the document consists of scanned images.

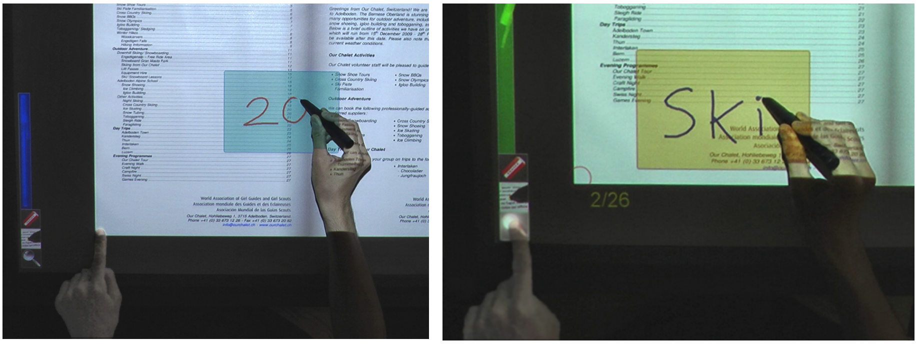


Figure 5.9: Handwriting a page number to turn to (left) or search keywords (right)

– Searching with handwritten keywords (Figure 5.9 right): users can perform text searches in the document by activating the magnifying glass icon in the command sidebar and writing keywords anywhere on the surface. When the finger is released, the handwritten keywords are recognised and the results displayed in a horizontal bar at the bottom of the screen along with the number of hits over the whole document for the recognised terms. Matches in the currently open page(s) are highlighted, while hits located on non-visible pages appear with contextual snippets and the number of occurrences on the referenced page in tappable sticky note-style markers at the side of the pages, as introduced previously (Figure 5.3 left). If the number of markers exceeds the space available at the side of the document page, navigation arrows appear at the bottom to cycle through the next or previous set of markers (this solution, that web search engines have adopted and kept for years, is a compromise between information visibility and interface clutter. Alternative methods, such as reducing the size of the bookmarks or stacking them sideways, would increase the former but also the latter). In the overview screen, terms matching search keywords are shown highlighted on all pages, although text might not actually be readable if the thumbnails are too small.

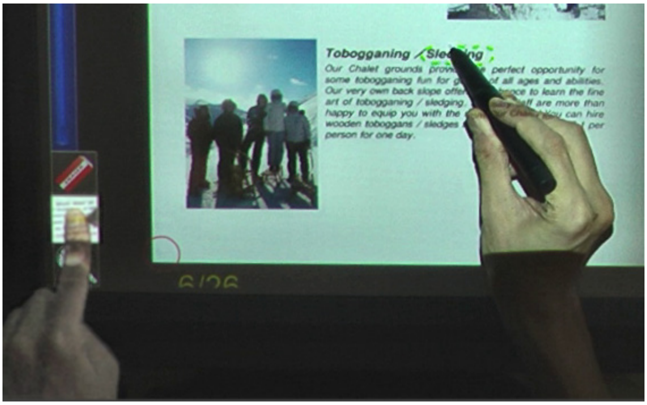
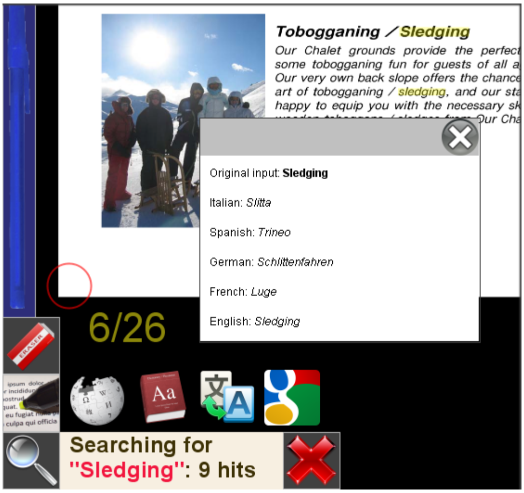


Figure 5.10: Selecting keywords inside a document with activated term-highlighting function

– Searching with terms selected in the document (Figure 5.10): keyword queries can also be executed by simply circling terms in the document. For that, the user activates the text marking function in the side bar and highlights text in the document. The lasso marquee appears as a dotted line to differentiate it from normal inking mode. In terms of number of chunks, this selection technique is equivalent to that of GatheredReader (except that one finger is required here instead of two to switch to selection mode), but it does not restrict the selection scope to entire lines. This also means that users can highlight text at different locations in the document without having to move their NDH at each time to frame the relevant passage.

As a later addition to the system, the possibility to use the selected or input keywords to query external sources and reference services, such as dictionaries, encyclopaedias, translation and web search was included (Figure 5.11), as consultation of external material is often necessary in AR.



Tobogganing / Sledging

Our Chalet grounds provide the perfect
some tobogganing fun for guests of all a
Our very own back slope offers the chance
art of tobogganing / sledging, and our sta
happy to equip you with the necessary sk
Our Cha

Original input: **Sledging**

Italian: *Slitta*

Spanish: *Trineo*

German: *Schlittenfahren*

French: *Luge*

English: *Sledging*

6/26

Searching for "Sledging": 9 hits

Figure 5.11: Using keywords to search in external reference services

5.2.4 Software Implementation

The application is developed on top of the framework introduced in Chapter 3. The class behind the main UI element representing the opened document is an extension of `InteractableComponent` along the model of the `ModeCanvas` presented in Listing 3.3. Thus, the component inherits basic navigational behaviour to move, scale and rotate the virtual document on the workspace. The rendering of the pages and their animations are implemented in the `paintComponent` method, where the geometric transformations are automatically handled by the superclass. The component also integrates moded pen strokes, whose appearance and function are defined in corresponding modes and associated to their interactional triggers (here, a button in a toolbar) via the `ModeConnector`.

Text search and target page turn both trigger an overlay, which is a `PenNTouchComponent` that is always on top. It receives the pen strokes input by the user and feeds them to a handwriting recogniser, Vision Object's MyScript [22], to convert it to machine-readable text. The recogniser module is integrated in the `GestureManager` and the text conversion is defined within a `Gesture` object associated with the two corresponding modes so that when recognition occurs the result is sent to the listening document UI component. The delete function is implemented in a similar way. When the document component receives the `gesturePerformed` event, it calls the gesture's `getPointerTrail` method to obtain the gesture stroke path, with which it can determine the intersecting annotations that should be removed.

Finally, for the two non-navigational gestures (multiple page flip and chop-and-spread) two special modes are defined based on the respective touch patterns of those interactions, so as to allow for clear separation of the processing logic in the touch event methods.

5.3 User Evaluation

5.3.1 Study Design

The main questions that motivated the design approach of the study were the following: would people's individual interaction practices and behavioural habits for paper and computers successfully merge when using the pen and touch tabletop? Could they be more, or at least as efficient on the tabletop as on the medium they would otherwise naturally use to execute a given AR task? Would the provided tools and gestures be sufficiently accessible and easy to master so that users could become rapidly productive with the application? To address those interrogations, a set of experiments was created with several tasks relying on different characteristics and inherent strengths of the considered media. For instance, paper was expected to prove more adequate than a regular computer for annotating, but the latter would most likely outshine the former in a task that involved much text searching. The question was then to see if the tabletop including both pen interaction and search functions could combine those intrinsic advantages and prove to be efficient in all cases. Those considerations led to a methodology with a series of tasks focusing on different aspects of AR separately rather than the summarisation exercise used by O'Hara and Sellen and Morris et al. [145, 148], hence the following protocol:

The study was comprised of three sets of data-searching and annotating tasks performed on three different media: paper, Adobe Acrobat on a regular desktop PC and the tabletop. The display for the desktop PC was a 19" monitor with a resolution of 1200 × 1024, whereas the projected image on the tabletop had a resolution of 1600 × 1200, as it was estimated that a higher resolution was needed for the tabletop to compensate for the optical distortions introduced by the Anoto dot pattern.

The tasks were designed so as to reflect scenarios where information needs to be located in a particular document (in this case a PDF) following which comments are added, for example for correction or summarising purposes. Participants were timed when performing each assignment and the approach taken to solve it. The manner in which they interacted with the documents was also observed for later analysis. On the tabletop, the number of times users triggered the different gestures and functions was recorded for each task. Counterbalancing measures were applied by rotating the orders of documents and media between users in order to reduce bias and learning effects. Like Morris et al. [145], a within-subject design was chosen so that testers could manipulate and directly compare the competing platforms in their feedback.

The study involved the participation of 20 people, 8 males and 12 females, who were a combination of students (12 persons) and members of the staff of the university (8 persons). The ages ranged from 21 to roughly 60 years old. All participants rated their ability to search for information on the Internet using keywords as good or very good. A majority also revealed they owned a touch-based device on which they read text content. As for their familiarity with PDF documents and Adobe software, most participants declared they used only the basic functions of the Reader to read and search within PDF files. A few testers had also tried the Text Edit tools in Acrobat, but none of them were proficient users.

The first 2 persons (2 students) were used as pilot testers, whose feedback was utilised to make a few adjustments to the study protocol that was then followed to perform the actual study with the remaining 18 people. Prior to the evaluation, participants were given adequate training on Acrobat and the tabletop with a focus on functions they were likely to need in order to fulfil the given tasks. During the execution of the tasks however, testers were left to decide which tools they preferred to employ and were not interrupted, except in cases where it was evident they had misunderstood the question.

In a nutshell, the evaluation consisted of three independent within-subject studies, each performed on all three media (i.e. three levels for the variable), that is, each participant had to execute 3×3=9 tasks in total. A session with a participant lasted for 2 hours and 10 minutes on average.

Both questionnaires and task questions used for this study are provided in Annex A, at the end of this document.

5.3.2 Description of the Tasks

Visual Search

A set of 6 documents with roughly the same number of pages was laid out on a table and on top of each was placed a piece of paper with a question, whose answer had to be looked for inside the document. The questions referred to items easily identifiable with a glance of the eye, that is, photographs, illustrations, headlines, captions etc. so that participants were able to find the answers by simply flipping through the pages of the document without taking the pain to peruse the actual content (e.g. "Where is the ad for the tour operating company that advertises hotels in Abu Dhabi?" in a travel brochure or "Where are the objects that may not be taken on a plane?" in an airport guide). The idea was to determine whether people adopt different strategies when trying to casually glean information from paper documents compared to digital ones and whether gestures like the chop-spread to trigger the document overview and the two-finger multiple-page flip would feel natural enough for the users to take advantage of. On a performance level, the average task completion time was expected to be comparable for the 3 media with perhaps a slight edge for the tabletop thanks to its overview functionality that could help spot relevant information more rapidly.

Text Search

A 20-page compilation of news articles was given to the participants with a list of 5 questions to answer in order. This time, people were told that answers were located in the text of the articles and so they had to read or at least skim through the content to find the solutions. Because of the tediousness of such a task when performed on paper, however, the questions were formulated in such a way that the corresponding news article could be easily determined from them (example questions were "What was one of the most popular names given to male babies in Switzerland in the 1990s?" or "How many deportation flights from Switzerland were there in 2008?"). On the digital platforms, of course, testers could avail themselves of the text search functions in order to directly locate the relevant passages and sometimes even the answer if the keywords were judiciously chosen. It is evident that here, paper was at a significant disadvantage and so the goal was rather to see how the tabletop would fare compared to Acrobat. In particular, it was interesting to observe how users would combine the navigational features of the two systems to manually scan for the information with keyword-based searching using handwriting or typing.

Editing

Participants were handed the news articles again and given 7 annotation tasks to perform. Those included circling, crossing out and marking items using (regular or digital) pens of different colours (e.g. "Circle in red all dates on page 15"). The last 2 tasks required users to go back to annotations they had previously made, this in an attempt to simulate a short review of the modifications they had done in the document (e.g. "Draw a vertical line next to the text where you made the circles"). For the latter, it was expected that the "Comments list" feature of Acrobat [167] and the annotation bookmarks of the tabletop application would come in handy, but overall, that users would prefer paper and the tabletop to make pen annotations.

Through the execution of those tasks, participants were made to realise the advantages and disadvantages of the tested media as well as their suitability to perform those tasks. Participants were asked to report on their impressions and provide feedback in a questionnaire after they completed the tasks.

5.3.3 Results

The study delivered much of the expected results, as the tabletop application proved efficient and pleasant to use by a large majority of the participants, despite people’s initial unfamiliarity with the system. With only minimal training, users were able to learn most of the gestures and immediately apply them to solve the given tasks. The bimanual Touch + Pen gestures, in particular, were quickly adopted, as users did not find it unnatural to use the pen on the same surface for inking as well as for command writing and to quickly switch between the two modes by activating a function with the other hand. One participant likened the operation to using keyboard shortcuts on the computer (an interesting analogy that is further explored in the next document application described in the following chapter). Generally, users liked the larger surface of the tabletop, especially in the overview mode, where they could easily have a bird's eye view of the content of the document as well as a perception of its length. They appreciated the freedom to move the document around the surface and write on the pages just like paper on a desk. There were also a few frustrations reported, which are detailed below.

The study also revealed a number of interesting findings concerning the ways people engage with paper vs. digital documents. For instance, people tended to stick to a particular approach on all media. Because paper documents were printed on both sides and were laid out in piles, participants quite naturally adopted a sequential searching method, where each page was viewed one after the other in order. On the digital platforms, however, despite the availability of multipage layouts or overviews, a few users somewhat surprisingly and counter-productively still set about the visual search task using the same method on Acrobat and to a lesser extent on the tabletop, by viewing one page after the other. Another observation was that initial conditions had a noticeable impact on the way a task was performed, in that most users did not change the initial viewing settings of their reader even in cases where the latter were not optimal. For visual search, the zoom factors of Acrobat and the tabletop were set so that an entire page or double-page could be displayed on the screen, while the text remained perfectly readable in both environments. This allowed users to quickly view the pages with minimal clicking, scrolling and finger-dragging. For text search, however, documents were deliberately set to appear larger so that users had to use the zoom functions in order to recover the optimal size. This, however, was not systematically done and in many cases, people were seen making heavy use of the mouse wheel to navigate to different parts of a page and the finger to move the document on the tabletop to access the side bookmarks when not immediately visible. This behaviour suggests that some type of automatic adaptation of the display context might be necessary.

Platform Efficiency and Task Suitability

	Paper	**Acrobat**	**Tabletop**
VS	*M=200,SD=67*	*M=172,SD=53*	*M=174,SD=73*
TS	*M=394,SD=99*	*M=296,SD=121*	*M=294,SD=91*
E	*M=346,SD=93*	*M=454,SD=133*	*M=331,SD=81*

Table 5-1: Mean execution times in seconds with standard deviation for the three tasks Visual Search (VS), Text Search (TS) and Editing (E)

Measurements of the completion times for the three tasks are reported in Table 5-1: Mean execution times in seconds with standard deviation for the three tasks Visual Search (VS), Text Search (TS) and Editing (E). A one-way repeated measures ANOVA (with the sphericity condition verified by Mauchly's test) revealed that while for visual search users were not significantly faster on any of the three platforms ($F_{2,38} = 1.71$, $P = 0.19 > 0.05$), statistically significant disparities were observed in the other two tasks ($F_{2,38} = 31.91$, $P < 0.0005$ for text search and $F_{2,38} = 13.01$, $P < 0.0005$ for editing). Post-hoc pairwise comparisons confirmed what is already apparent in the table, i.e. that the last two tasks were executed at least as fast on the tabletop as on the next best medium and faster than on the third (all p values < 0.007 for significantly different mean execution times). For text search, users could take advantage of the appropriate functions on Acrobat and the tabletop in order to almost instantly locate the relevant contexts of the sought information, something that was of course not possible on paper. Furthermore, people were able to be almost as efficient handwriting keywords as typing them with a keyboard. The few occasional recognition errors were indeed compensated by the availability of term highlighting and page markers with context snippets to help find the required information.

For the editing tasks, participants found themselves in familiar territory with the digital pen, with the added benefit compared to its felt tip counterparts that mistakes could be easily deleted. Acrobat's commenting tools proved too cumbersome and understandably users felt more at ease with regular or digital pens to mark and annotate text. Those feelings were confirmed when study participants were asked to rate on a Likert scale their overall experience with the three media as well as for the three individual tasks, particularly with respect to the suitability of the medium to execute said task. Those results are reported in Figure 5.12.

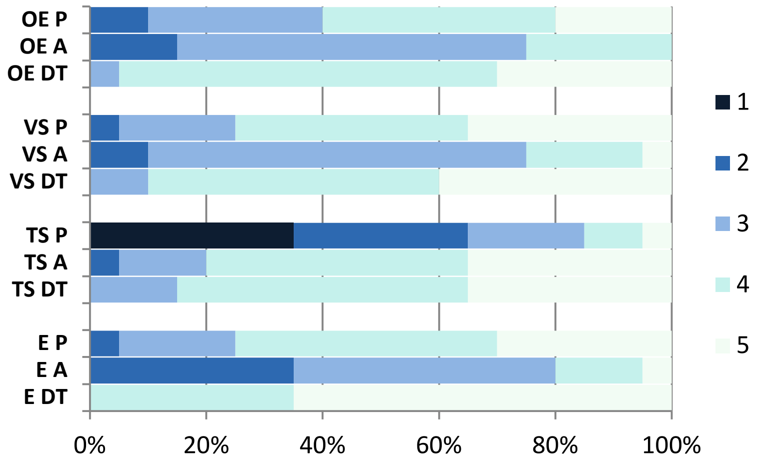


Figure 5.12: Participants' ratings of their overall experience (OE) on the three media and the latter's suitability to perform the given tasks from 1=Very Poor to 5=Very Good.

Friedman tests performed for all four comparisons exhibited statistical significance in each case so post-hoc Wilcoxon signed-rank tests with Bonferroni adjustments were made to determine pairwise significance. For overall experience, the tabletop was rated significantly higher than Acrobat ($Z = -3.493$, $P < 0.0005$) and even paper ($Z = -2.524$, $P = 0.012$). For the visual search task, participants found that the tabletop was significantly more suitable than

Acrobat (Z = -3.508, P < 0.0005), but not with respect to paper (Z = -1.076, P = 0.282). In text search, the tabletop was rated equally suitable as Acrobat and both significantly higher than paper (P < 0.0005). For editing, the tabletop was rated as more suitable than Acrobat (Z = -3.831, P < 0.0005) and even paper (Z = -2.754, P = 0.006), which was somewhat surprising. User feedback revealed that while they still felt more comfortable marking paper, they valued the additional tools provided by the tabletop application. Two users commented that they imagined in a real situation they would want to store their annotations digitally and so they welcomed the potential benefit of being able to do that on the tabletop.

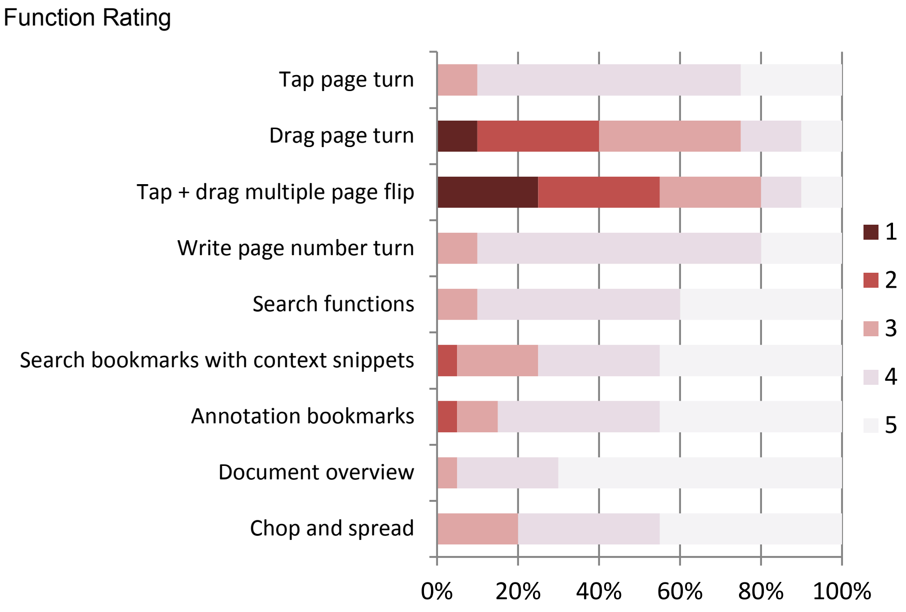


Figure 5.13: Users' ratings of the usefulness of the different gestures and functions ranging from 1=Not useful at all to 5=Extremely useful.

Although participants were instructed, prior to the task assignments, in the use of all gestures and functions of the tabletop without any particular bias for any of them, all interactions did not enjoy equal popularity. The post-task questionnaire requested testers to assess the general usefulness and practicality of those features, not necessarily in association with the given tasks. The answers they supplied (captured in Figure 5.13) very closely matched the frequency in which they used the different functions, as logged by the system. For instance, users found the overview feature and its associated gesture very helpful and hence used it extensively, whereas the bimanual multiple page flip, on the other hand, saw little appeal and so was almost never used. This is most likely due to the difference in complexity and precision required by the gestures, as the former essentially executes a transition from one state to another and can be triggered anywhere on the surface using a coarse gesture, whereas the latter maps a precise position of the finger to a particular visual state (the multiple pages being viewed and their folded appearance as they are being turned). However, further observation of selected test users, who spent longer time with the application, hinted that this discrepancy could also be put down to learnability and awareness issues. Some gestures simply take more time to master than others.

System Limitations and Discussion

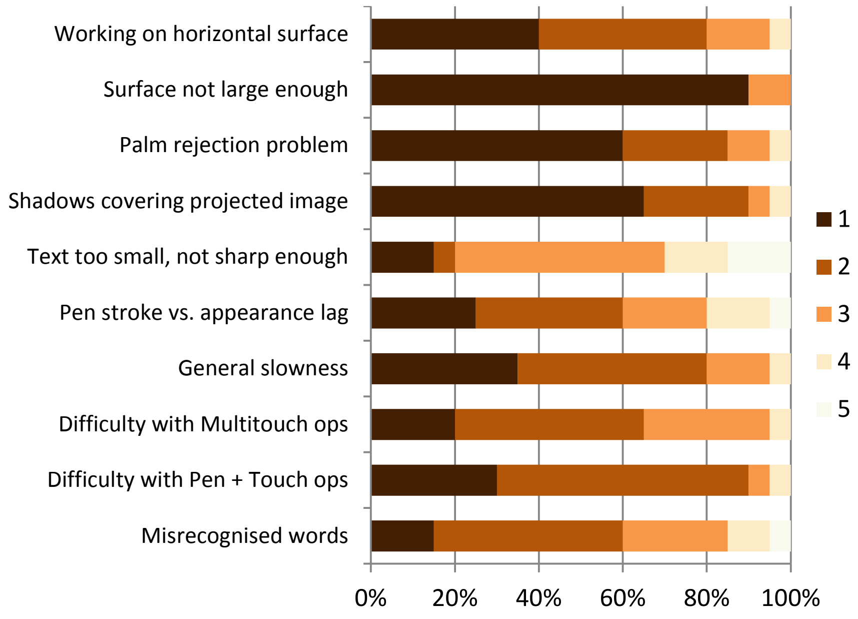


Figure 5.14: Hardware and software limitations of the system and how seriously users felt their experience was affected by them, ranging from 1=Posed no problem at all to 5=Extremely problematic.

Due to a number of hardware and software constraints as well as design choices, the application suffered from a few limitations that more or less interfered with the execution flow of the user's work. Study participants were asked to report on these impediments and how far they were affected by them. The results are presented in Figure 5.14.

Despite the higher resolution of the projected image on the tabletop compared to the PC monitor, the most serious hindrance was the insufficient sharpness and crispness of the text preventing pleasant viewing and reading of the document pages, especially in the overview screen. As hinted at in the system description section, this lack of adequate rendering quality is mostly due to the dot pattern on the overlay but also, to some extent, to the display resolution as well. This issue of resolution depth, both for display and input capturing, is critical and needs to be addressed by tabletop manufacturers if they are to support a wider range of applications requiring high-quality rendering of object details such as AR and generally document-centric applications. A possible yet more expensive solution for the viewing problem could be to utilise high-definition projectors or multi-projector displays [31]. As for sensing, Anoto technology, which was initially designed for asynchronous recording of pen inking on paper, has been adapted to support streaming position data, but because it relies on a printed dot pattern it is not ideal to integrate with touchscreen systems. Moreover, optical pens are still fairly bulky and need to be regularly recharged. The problem of high-resolution and high-accuracy pen input sensing for tabletop systems therefore remains.

Another reported problem was the misrecognition of handwritten keywords that sometimes caused a little frustration to some users. However, those misrecognition mistakes only occurred for a small number of participants, who had trouble adapting to the writing style required for the handwriting recognition engine to accurately determine the input text. With more training and better support by the interface, this handicap could most likely be eliminated or at least alleviated.

Among the factors that posed only minor disturbances was the signal interference of the resting palm or arm on the tabletop while writing. Participants in the study were told to pay attention not to come into contact with the touch surface when using the pen, but many of them did so naturally and so were not troubled by that constraint. The evaluation tasks did not involve much inking, which is also a reason why users might not have felt too inconvenienced. One can imagine, however, that more pen-intensive tasks requiring a great deal of manual text entry or sketching could cause wrist fatigue in the long run and so this problem merits consideration.

Finally, an often pointed out drawback of front-projected systems, namely the occlusions caused by shadows cast by arms reaching over the tabletop, turned out not to be an issue for the vast majority of the study participants. Shadows are natural occurrences that most people can cope with, especially if they can be controlled [183]. One interesting matter that was brought up by a left-handed user, however, was that shadows did get in the way when writing with a curled left hand (i.e. almost from above), since in that case the pen-holding arm and wrist cover the letters that have just been written as the arm moves from left to right. But all things considered, the occlusion problem of front-projected displays was found to be of minimal discomfort to the users and so the use of such systems should not necessarily be discouraged, at least in those kinds of usage scenarios.

5.4 Enhancements

The active reading application and the conducted user study were a first attempt to sound out the potential of interactive pen and touch tabletops in a document context. There are naturally many improvements that could be made to the interface, both in terms of design and features in order to support other AR requirements. A few possible avenues are:

- Blank "sheets" for clean annotations: users are currently only able to create annotations on existing pages of a document. Often, however, there is a need for a clean surface on which to write either because no free space is available on the page to annotate, or because the notes need to be taken away. To support this requirement, the application could include virtual sticky notes or sheets to be either tucked in the document or laid aside as separate material, much like physical stationery. This type of solution would be in line with the workdesk metaphor and is realistically feasible because of the large screen real estate afforded by a tabletop display compared to e-readers and tablets, which are more space-constrained (leading to the necessity of space-saving measures as in GatheredReader, where note sheets appear in a split-screen configuration with the document).
- Extraction of and association of notes with selected document content: the prototype only allows users to make freeform annotations and link them to their corresponding pages via dynamic bookmarks. Both LiquidText and GatheredReader support finer-grained content selection, extraction and back-linking in addition to penned notes. Such functionality could be integrated in the application, using perhaps mechanisms inspired by clipping-extraction techniques of other systems [94, 213]. Captured content could be "pasted" on the virtual note sheets mentioned above. Information would naturally still be attached to its source so that tapping on notes or pieces of extracted data would open associated passages in the original document.
- Support for multiple documents: AR sometimes involves several documents from which information is collated and collective notes are taken. Thus, the interface could simply allow several documents to be displayed and manipulated together, as on a real desk. Automatic layout strategies could also be included to deal with screen clutter.
- Note utilisation: systems developed for AR so far have only considered the note-taking and data-gathering aspects, and have not really dealt with how those notes are later utilised (with the exception of InkSeine [94], maybe, although technically it is not a system specifically designed with AR in mind). Strictly speaking, acting on notes is not part of AR, but since the purpose of the task naturally influences how it is performed by users, developers aiming to support it more effectively should consider it in its wider context. The type and manner in which notes are gathered might indeed differ if those notes are meant for oneself or if they are intended for other people, if they are simple markings of relevant passages for future reference or if they will form the basis of a more complex task(s) etc.

The integration of the first three features outlined above is relatively straightforward. In terms of implementation, those functions can be supported by further pen and touch interactions executed directly on the workspace or in combination with specific toolbar commands.

Regarding design choices for the user interface, it remained fairly close to the paper metaphor without strictly adhering to it when practical concerns dictated that it should adopt a more straightforward approach to support useful functions (e.g. searching, overview mode). Yet, the user study revealed that, for some aspects, trying to replicate paper-like behaviour is not always rewarded with success. The multiple page flip gesture, as it was implemented, is an example where users found it added little to the program's functionality. The lesson one can draw from that when designing systems for pen and touch user interfaces is that there needs to be a balance between the desire to offer a close-to-real interaction paradigm to appeal to users' sense of familiarity and the necessity for the system to provide handy and effective tools to execute the target tasks. In particular, designers should not obstinately seek to model a physical interaction simply because it exists in the real world and therefore think it should be recreated somehow on a virtual interface, even though the latter is not necessarily well adapted for that.

In terms of user validation, the conducted evaluation compared the tabletop application with fairly traditional platforms that have clear disadvantages with regard to some of the tasks. Either searching is difficult (paper), or editing/content marking is (Acrobat). What the study has not shown is the advantages of digital tabletops over the other major contenders: tablets and e-readers. So far the pros and cons of those two different categories of devices have not really been thoroughly quantified in this context. Studies to identify what kinds of benefits a large tabletop surface can bring to AR activities compared to mobile devices with more limited screen space (and conversely what they sacrifice because they are not portable) remain to be done.

With the experience and teachings of this first experiment, the next step towards tackling more ambitious document-engineering tasks can be undertaken. So far, techniques to manipulate documents have been limited to navigation, basic annotation and searching, i.e. operations that do not change documents' content or layout. While document-editing features could be added to the AR prototype and the system evaluated anew, it is more interesting (and perhaps more sensible) to start from a clean slate, not only to be able to avoid the flaws of the current system and optimally adapt the interface to the new requirements, but also from a research perspective to try another approach that yields different kinds of results.

The next chapter explores the pen and touch input paradigm in more depth and presents the next prototype, which supports document creation and editing.

6 Document Scenario 2: Document Authoring

The previous scenario, active reading, involved only limited document editing capabilities, as only freeform annotations could be added and content could not be directly modified. As motivated in the introduction, the objective of this thesis is ultimately to show that a pen and touch tabletop environment can support advanced document editing tasks as well, including authoring of high-quality documents from scratch. Naturally, this entails a significant increase in functions and features to integrate. The challenge is therefore to leverage pen and touch interaction techniques for the realisation of authoring operations and manipulations with which users are able to produce rich, professional-looking electronic documents. In particular, the possibility to map commands to adequate gestures instead of resorting to explicit widgets and UI controls is worth investigating, as suggested by the results of the third and fourth experiments of Chapter 4. The AR prototype introduced practical examples of NDH-based pen mode switching to perform non-inking document actions. Can this paradigm be applied in a larger scope to support an extended range of editing functions? If yes, what kinds of processes can be efficiently performed with gestures and when would one be obliged to fall back on more traditional WIMP techniques?

One powerful analogy that can be made with the traditional desktop computing world in this context is that of keyboard shortcuts, in the sense that a particular mode set by the NDH, within which the pen-holding DH articulates a variety of actions, is very similar to commands triggered by combinations of modifier and regular keys on a keyboard. As immediately accessible triggers, the two types of modifying techniques are natural extensions of the normal keyboard typing or pen inking mode. Contrary to modifier keys, however, modifying NDH interactions need not be constrained to a particular location, which allows blind activation, the advantages of which have been demonstrated by Experiment 4. Furthermore, prior work suggests that stroke-based shortcuts have cognitive advantages over keyboard shortcuts [29], which means that the learning curve is potentially much less steep, compared to traditionally complex desktop office applications, for which productivity is often associated with the ability to master keyboard shortcuts [117].

If documents are to be fully editable and those documents contain text (as is the case of many office documents), then the question of text entry inevitably arises. Previous studies show that text input is one of the major roadblocks to the adoption of tabletops as efficient work systems [37, 166]. While possible solutions to tackle that problem have been proposed [95], none are totally satisfactory for horizontal interactive screens. Physical keyboards would likely be the most efficient, but they are also cumbersome and frequent switching between text input and other tasks that involve direct touch on the surface can quickly become tedious. Soft keyboards and styli are more flexible lightweight instruments but the same level of output cannot be achieved with them. Rather than attempting to support all types of documents for which typically there is an array of dedicated applications within office suites, it seems wiser to play to the strengths of the tabletop and pen and touch interaction. Hence, one might consider documents with strong layout requirements and graphical content rather than with large amounts of text (without forsaking support for text entry either).

The structure of this chapter is the following: to begin with, the design approach is laid out, where in a first part, the pen and touch input paradigm is taken up again and an extended gestural model is formulated, according to which the non-dominant hand defines different pen modes through modifier postures, in analogy with modifier keys of a keyboard. Then, the scope of the application prototype is defined, in particular, target document types and editing functionality to support and how this should be achieved. In the second section, the document model underlying the authoring environment is introduced. Based on this theoretical framework, a range of gesture-driven document-editing interactions are presented, together forming a coherent document composition and editing platform. In the last part, the qualitative results of a lab study are reported, which help identify the pros and cons of the approach.

Part of the material of this chapter was previously published in the adjunct proceedings of the CHI 2013 conference [143] and the proceedings of the ITS 2013 conference [141].

6.1 Design Approach

6.1.1 Gesture Model

As has been shown in the literature and with the previous prototype, a pen and touch environment does not preclude unimodal interactions. Classic unimanual multitouch manipulations to pan, zoom and rotate objects are naturally still possible and widely utilised but so are bimanual multitouch commands, as the pen can be nimbly stowed away in the DH during the execution of such operations [85, 142] (see also section 2.4). The stylus, when used on its own, performs mainly inking actions and gestures, such as lassoing to select and scratching to delete objects [75]. The input space of combined pen and touch interaction is therefore an additional layer that comes on top of those existing unimodal gestures to extend the total interactional vocabulary available to the UI designer when mapping functions to hand gestures.

Figure 6.1 illustrates the bimanual gesture design space with its different parameters available for the creation of compound pen and touch interactions. Essentially, the building blocks for those combinations depend on two factors: location and type, which each apply to the NDH and DH, thus yielding 4 dimensions or degrees of freedom (and an optional 5th if NDH motion to control further parameters is allowed). The posture and gesture type components can be used to form a grammar of two-handed transactions. For the NDH, a posture type is a chord [199], i.e. a combination of fingers [85, 140] or any distinctive shape formed by the contact area of the hand on the surface that can be reliably identified [200].

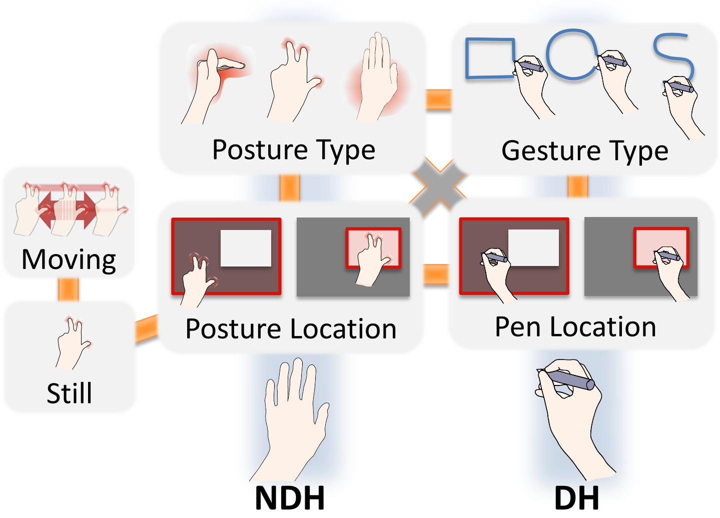


Figure 6.1: Bimanual gesture design space showing different axes across which combinations of unimodal interactions can be constructed

Similar to modifier keys on a keyboard such as Control and Alt, NDH postures can be viewed as touch-based modifiers, which, together with pen gestures, provide shortcuts for a wide vocabulary of commands. Typical stylus gestures are lines, circles, rectangles, ticks, pigtails etc.

The position factor allows source and target objects to be specified for interactions. Thus, a gesture with an NDH posture initiated on a particular UI element can have a different effect compared to if it is applied in a general context, i.e. on the whole workspace. Similarly, a pen gesture may be interpreted differently, depending on which object it is executed on. Thus, one can distinguish global postures that control the general pen mode and can be triggered anywhere on the surface, from local or location-constrained postures that apply to the specific area or component on which they are performed (e.g. finger hold + pen drag gesture [75, 93]). With global modes, the target of the manipulation is either determined by the pen (for instance the first or last pointed object) or it is a general, application-wide operation that affects the whole interface. Local modes on the other hand allow two pointing parameters to be set, one with the NDH and the other with the DH. This can be used, for example, to determine a source and a target or to specify a range. In both cases the user is able to remain concentrated on the DH's task at all times, as in the first case, they do not have to pay attention to where the NDH is located to execute the posture (blind switching) and in the second case, the NDH operates in the vicinity of the DH so the locus of attention is the same. This approach differs from Hamilton et al. [85], who, even for global postures, require that they be executed on a special command panel at the bottom of the screen. Hence, instead of setting a single inking property as in the fourth experiment of Chapter 4 (stroke style), modifier postures can be general controllers of whole categories of non-inking pen functions.

To give the user additional control possibilities, one can further allow posture movements such as lateral dragging or "scrubbing", as in [85]. However, because this adds a further chunk to the gestural phrase [54], further hand coordination is required and so care must be taken not to cognitively tax users when designing such compound interactions.

To summarise in semiotic terms, posture and gesture types control syntax, whereas locations determine the pragmatics of the interactions [108].

Related to blind mode setting is the issue of maintaining awareness of the currently active mode via appropriate feedback mechanisms. In the bimanual coordination experiment (Experiment 4), a simple icon was used, which may be adequate for local postures, if shown in the area of action, but, as participants of the study remarked, is easily overlooked in the case of global postures, where pen operations can occur anywhere on the screen. At the same time, one should be careful not to choose too large visual indicators, as this would be too disruptive for the user. Therefore, an appropriate compromise needs to be found in order to gracefully capture the user's attention, without interfering with their main task.

Finally, because global postures, by definition, can be triggered everywhere they can potentially collide with other unimodal gestures and basic multitouch interactions, especially one-finger panning and two-finger scaling. A simple solution to avoid such conflicts is to choose postures based on three or more fingers or large contact areas.

6.1.2 Scope Definition

Target Document Types

One of the major questions to address first is to consider what types of documents might be more suitable for the pen and touch desktop platform. As mentioned in the introduction, one can intuitively surmise that page-based documents with much graphical content or assembled from various elements are more fitting candidates than continuous, text-intensive documents, for which the PC and physical keyboard are arguably still the best tools. Hence, in this prototype effort, the design of the interface and interactions should be geared primarily to documents of the former category, e.g. presentations, flyers, brochures and similar documents typically authored in presentation software and DTP environments. Narrowing the target range to a particular kind of documents also allows the design approach to be more focused, so that the resulting interface does not become too complex and thus unwieldy, in attempts to support all the specifics of diverse types of data.

In light of the above considerations, the typical document to be authored using the proposed system contains a mix of styled text, vector data (shapes) and raster images. Those elements are laid out on individual pages as a structured collage, i.e. the spatial arrangement is not flow-based as in a word processor, but is done according to absolute positioning, as in a presentation creation tool. Complex interlinking between elements, such as connected text boxes for text sequences spanning several frames as in DTP design, is not considered at this stage (although this may be an interesting option to investigate in future work, as previous projects have shown the potential of pen-based interfaces for diagram-like editing [75, 139]).

Editable Document Element Properties

A quality of advanced document authoring software is that almost every possible element property can be edited and changed. Text can thus be styled in a number of ways, from font type and size, appearance, alignment, foreground and background colours etc. For shape elements, in particular, users expect to be able to modify attributes such as stroke colour, width and style even after the shape has been drawn. Therefore, it is not sufficient to simply detect shapes, strokes and their features (e.g., if the line is continuous or dashed) on inking. The interface has to provide means to conveniently edit those properties after input, without requiring the element to be entirely redrawn.

Reality-Based Interaction

As has been established, pen and touch interaction affords a broad expressional vocabulary that can be drawn upon to support a wide range of application functions. The potential to implement commands through a set of uni- and bimodal gestures theoretically allows to significantly reduce reliance on widgets and other UI control artefacts that typically clutter interfaces. As many NUI-designers have identified, interfaces should leave as much screen space as possible to user content in order not to disrupt the execution of the task at hand [116].

Jacob et al.'s Reality-Based Interaction (RBI) framework [99] provides a theoretical foundation for the design of post-WIMP interfaces, which can be applied to pen and touch tabletop applications. This model, which recognises the increased role of real-world skills and practices in the design of modern interfaces, is especially relevant for a document-authoring system, as people can potentially bring their experience of engaging with physical documents into the digital world. The challenge, therefore, is to try to take advantage of those skills, when possible, by designing interactions that are either directly imported from real pen-on-paper experience or based on powerful analogies. Those RBIs, together as a cohesive toolset, should enable users to efficiently produce structured electronic documents similar to those authored using office and DTP software.

The design process therefore considers what types of functions can be achieved through gestures and when helper tools are unavoidable (e.g. to select a colour or a particular font). The goal is to create an environment with a mostly natural pen-on-paper experience, but where the output is high-quality digital documents. The system description below explains how those various challenges are addressed, following those principles.

Document-Editing Features to Support

To meet the requirement to fully support the authoring of documents, as defined above, a minimum set of tools and features to include in the first iteration of the application can be defined:

- Document creation from scratch: users should be able to create new documents from nothing using multiple formats.
- Document navigation: as in the active reading application, it should be easy to navigate within a document using appropriate interactions. As an improvement on the previous prototype, the editor should handle multiple documents simultaneously in order to enable inter-document manipulations.
- Element manipulation: operations to add and delete document elements (including pages), as well as to copy and move those elements within and between documents should be available.
- Text input and styling: text entry is performed using handwriting recognition as explained above. In addition, basic text styling options should be present, including colour, font type (family), size and style. Furthermore, it should be possible to determine a paragraph's horizontal alignment: left, right, centred or justified.
- Shape design: similar to text, shapes are hand-drawn on the editor canvas and vectorised using an appropriate recogniser. A number of styling functions can also be integrated, here as well.
- External content retrieval: for presentations, brochures and other illustrated documents, authors often seek to enrich their content with professionally designed graphics, art, symbolic drawings and illustrations etc. collectively referred to as cliparts. Efficient retrieval and insertion of such elements should be supported by the system.

There are naturally many more functions that could be included and document element types that could be supported (diagrams, tables etc.), but the goal is not to build a full-fledged document authoring product. Hence, the scope of features to integrate is deliberately limited to this core set of items, which are deemed sufficient to build a working prototype for meaningful research analysis.

6.2 Application

6.2.1 Workspace

As with the AR prototype, the setting of the interactive workspace follows a desktop metaphor, where documents appear on a virtual work desk and are directly manipulated by users via multitouch and pen input (Figure 6.2). The interface shows all open documents, by default a single page of each. Common drag, pinch/spread gestures to pan and scale documents are available. Documents can be expanded and collapsed to respectively display and hide other pages using a single-finger shaking gesture (Figure 6.3), as if scattering real pages from a pile. Expanded pages can be laid out horizontally or vertically depending on the direction of the shaking motion. When pages are collapsed, the document appears as a pile with the currently selected page on top of. A double tap on a particular page triggers an instant fit-to-screen zoom to enable more precise content editing. Instead of repeated page-dragging, lateral swipe gestures can be used to quickly flick from one page to another. Following the design objective to maintain a smooth experience without tedious context switches, document pages are always editable, regardless of zoom factor or state, i.e. users can input and manipulate content on pages whether documents are collapsed, expanded or magnified.

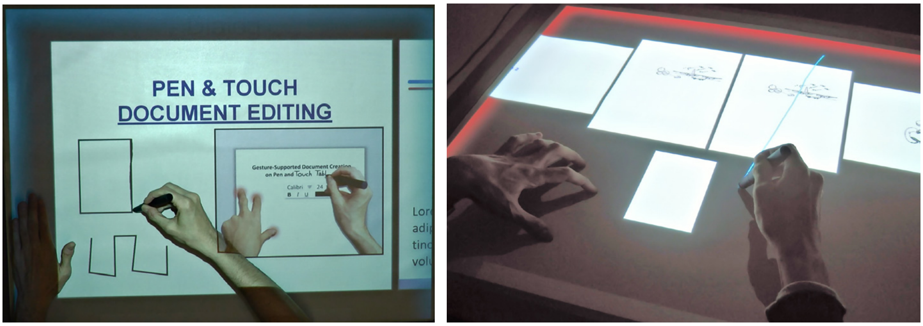
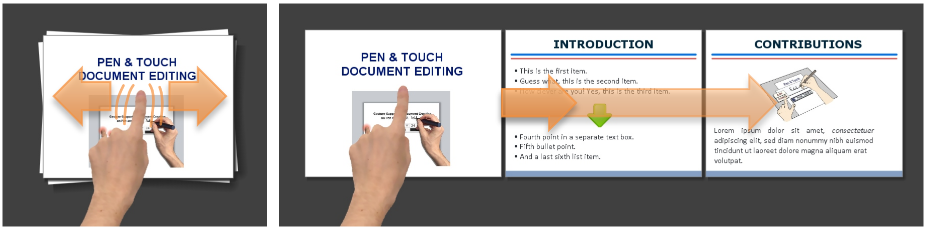


Figure 6.2: The document-authoring environment



• This is the first item.
• Guess w…t, this is the second item.
• …you! Yes, this is the third item.
• Fourth point in a separate text box.
• Fifth bullet point.
• And a last sixth list item.

Lorem ipsum dolor sit amet, *consectetuer* adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Figure 6.3. Finger shake gesture to expand and collapse documents.

Various page manipulations to move and copy elements within and between documents are supported: moving a page is achieved by placing three fingers on it (three fingers to distinguish the gesture from one-finger panning and two-finger scaling) and dragging it to a new location, inside an existing document or on the workspace to create a new document. Because the muscular tension imposed by a three-finger dragging operation is particularly demanding, the gesture can be relaxed so that the dragging motion can also be performed with a single finger after the move operation has been registered [208]. Pages can be copied instead of moved by holding them with a finger (local posture) and dragging them away with the pen. This gesture is similar to those described in [93] and [75], but here additionally, NDH and DH can point to different elements to indicate a range of pages to copy or move (Figure 6.4 left), thereby making use of the ability of local postures to determine two gesture parameters with parallel pointing actions of the two hands. Furthermore, if two fingers are used to hold down a page rather than one, a blank copy is created instead, as a convenience to add new pages with the same format (Figure 6.4 right). In the latter case, a quick pen swipe in the direction of where the page should be created is also sufficient to execute the operation. This is the equivalent of the "create new slide" function of presentation creation programs.

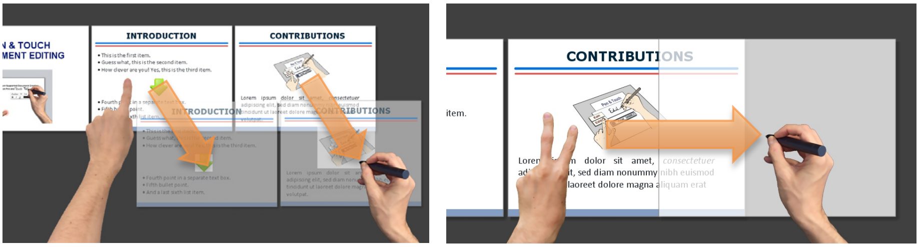


Figure 6.4. Copying a range of pages with one-finger hold + pen drag action (left) and creating a new blank page with two-finger hold + pen drag (right).

Document elements that make up the content of a page are independent components that can be moved with single-finger dragging within a page, between pages of the same document and between different documents. Resizing is performed using control handles located at the edges of the element's bounding box, as in the second and third configurations of the pen and touch experiments (see Figure 4.2 in Chapter 4). This choice of gestures allows users to pan and scale the entire document and move individual pages around even while actually touching an element, thanks to the chord requirements to perform those actions. Thus, to consistently target the entire document for a move or scale action, users can use two fingers, which will not affect any underlying element as transformation operations for the latter are performed using a single finger. The reasoning is analogous for page moves, which require three fingers for gesture registration.

6.2.2 Global Modes

For the document-editing purposes of the application, five different global modes are defined: two inking and three non-inking. The two types of inked input for the stylus are text and shape. Both data types are subject to on-the-fly analysis by dedicated recognition engines (VisionObject’s MyScript [22]) in order to convert user strokes into properly formatted content, respectively typeset text and smooth vectorised shapes. The default input mode, i.e. when the pen marks a page without NDH-modifying, is text. For shape and freeform strokes, the user has to indicate the mode with an NDH posture, specifically a flat hand, which mimics the holding of a sheet of paper (as in Guiard's example [82]) while drawing with the DH. As the detection of this pose relies on the total area of the contact bounding box, which has to exceed a high threshold to be activated, any further touch contacts do not cause the mode to change (as they only increase the bounding box). The side effect of this is that, as long as the flat NDH properly maintains the mode, all other touch input is ignored, thereby providing free palm rejection.

In the initial version of the prototype it was not required to explicitly set which type of content was being inserted, as the goal was to let users freely ink pages regardless of content type, as they would on a physical sheet of paper [143]. To determine how to convert pen data, an algorithm estimating the likelihood that input strokes were text or not was used. But early tests showed that in many cases the wrong data type was inferred (e.g. a letter 'O' instead of a circle, or a line instead of a hyphen etc.) and so rather than risking causing user frustration it was decided to strictly enforce the separation of the two kinds of content with different modes. Some level of automatic element segmentation is included, in that if the user draws a grid-like sketch, a table element is created in the document model instead of a shape.

The following assignments of postures for the three remaining non-inking modes are then defined. All postures rely on finger combinations or chords [199]: three fingers for content-modifying commands, four fingers for selection operations and five fingers for content-insertion shortcut gestures. Hereafter, those modes are referred to respectively as Command Mode, Selection Mode and Content Mode.

All postures are quasimodes [160], i.e. they are maintained throughout the pen action. In addition to providing kinaesthetic awareness as explained before, such a design has the advantage that an operation can be immediately cancelled during the pen movement without incurring any penalties, by simply lifting the NDH before the pen action has completed. From a performance point of view, using NDH postures to control how pen input will be interpreted means that only recognition engines that are necessary in the selected modes can be activated. Hence, handwriting recognition need only be active in Text Mode, shape recognisers in Shape Mode and gesture detectors in Command Mode. Clearly separating those interactional categories increases responsiveness and makes sure there is no overlap between the different gestures, without compromising the breadth of available expressions.

As for the awareness issue regarding visual feedback for the active mode, the solution adopted in this prototype uses light gradient borders surrounding the workspace with distinct mode-characteristic shades and matching stroke colours for the pen (Figure 6.2). Such a visual cue is easily spotted by the peripheral vision, without interfering with the main interface and the task at hand.

6.2.3 Stroke Input and Styling

Text

As stated above, text is entered through normal handwriting and converted on-the-fly by the recogniser, where the resulting typeset text appears *in situ*, i.e. it replaces the user's handwritten content. This more direct input method was chosen instead of a separate writing pad (see Figure 6.5 for an example), as such tools create a disconnection between the user's pen (input) and the ink shown on the screen (output), which detracts from an experience close to pen-on-paper.

A further adopted design guideline inspired by RBI sees that what can be reliably derived from pen input should be done so, while other aspects that are hard to accurately infer from hand-drawn content should be determined through explicit selections. Following that strategy, properties such as text size, underlining etc. which can be directly obtained from ink data are teased apart from characteristics such as font type and style that are difficult for users to imitate and so need to be specifically set. Thus, when text is entered in a blank area of a document page, the size of the font used for the recognised text is determined so as to roughly match that of the user's script. If, on the other hand, inked data is initiated close to or within another text element it is considered an edit operation and so the recognised result is added to the element and the font size and style of the nearest letter in this element is adopted, which is the usual convention in word processors.

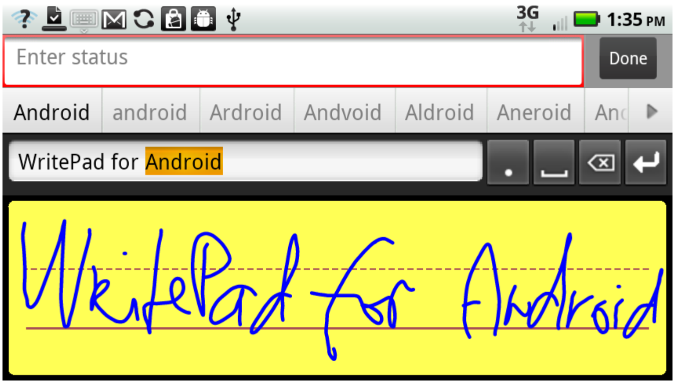


Figure 6.5: Phatware's WritePad for handwritten text input on Android devices [20]

The text input scheme also allows handwritten words to be inserted in the middle of text elements. The user simply needs to start writing within the element for a gap to open at the insertion point. As the user writes their text, the right part of the edited element continuously shifts to create more space (Figure 6.6). If a portion of the text is highlighted the user-inserted content replaces the selection. This blended text-editing technique is an attempt to recreate standard text-insertion behaviour with handwriting.

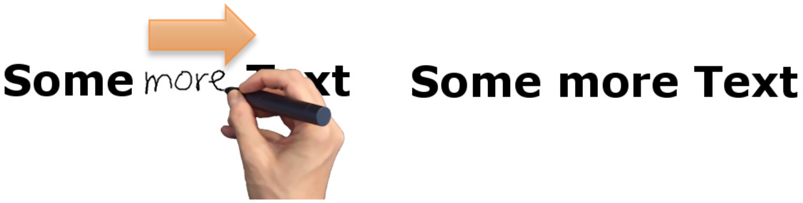


Figure 6.6. Insertion of handwritten text within text element. Text after the insertion point is gradually shifted to create space for writing.

Because it is not possible to insert isolated white spaces in formatted text elements using handwritten input, two convenience pen gestures in Command Mode are included for that purpose: a chevron stroke gesture (^) adds a space (a specialisation of the common text-insert mark used in proofreading) and a ↵ gesture inserts a line break.

In Selection Mode, text highlighting with the pen is performed in a manner similar to most pointer-based interfaces, i.e. by dragging the pen across the desired portion of text. Quick selection shortcuts found in modern word processors to highlight entire words and paragraphs with double and triple taps, respectively, are also available. As with pages, cut and copy paste of text are supported through respectively single-finger touch dragging and NDH touch-hold + DH pen-dragging. The selected text can then be touch-dragged within the element to be placed at another index. If the cut/copied text portion is dragged outside the bounds of the element and released on free page space, a new text element is created.

The interface features a set of "chopping" gestures to perform alignment operations on text elements. Those shape-based interactions allow users to set the horizontal alignment of a text element based on how the gesture contact shape intersects the element's bounding box. Single chops close to the element's left or right borders align the text accordingly left or right, whereas a chop roughly in the middle of the object centres the text. Finally, a double-chop with the two hands on both lateral edges of the element's bounding box causes the text to become justified (Figure 6.7). Those simple gestures have the advantage that they combine scope framing (target text range) and operation execution (alignment) in a single chunk [54], which differentiates them from fixed guides and rulers such as rails [200] and the composite and dragged gestures of NEAT [77].

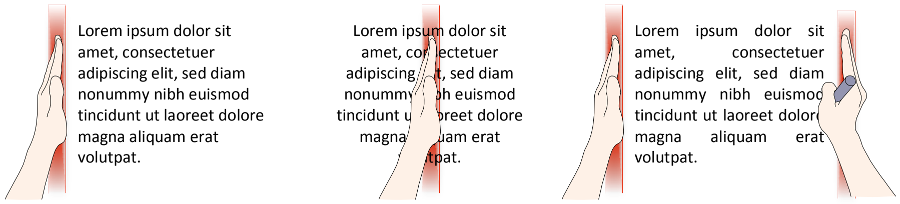

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Figure 6.7. Chopping gestures for text alignment. Illustrated are left, centred and justified alignments.

Except for underlining, which is directly drawn under the desired text with the pen, styling options for stroke data are set via a pop-up toolbar (Figure 6.8) summoned by a single-finger touch action, a short-press if initiated on empty desktop space and a slightly longer press if activated inside a document (in order to distinguish it from move operations, as with the in-place popup menu design of Experiment 3 in Chapter 4). The toolbar also automatically appears on a text-highlighting action just below the edited text element. The style properties that can be set in this manner are font family, colour and font style (bold, italic and normal). Here both touch and the pen can be used to select style options, the second alternative being perhaps the most effective when the pen is close to the widget (as in the text-highlighting case) or when several selections need to be made in succession. To maintain user awareness about the currently active font and style options at all times, the font family name styled according to the selected options is displayed semi-transparently at the top of the interface.

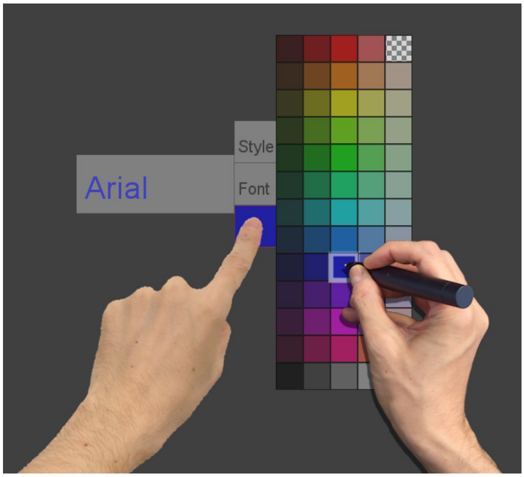


Figure 6.8: pop-up toolbar with style options. The NDH finger slides on the style category and the pen-holding DH selects the parameter

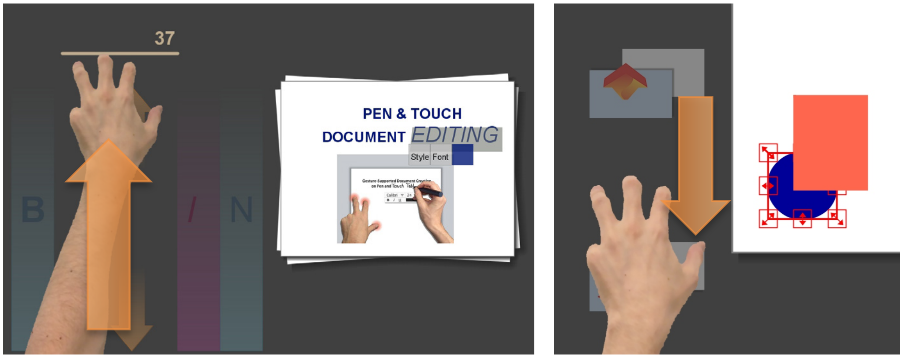


Figure 6.9: NDH vertical dragging gestures in selection mode: changing text size (left) and element Z-order (right)

Font style as well as size changes can further be set using an interactive overlay controlled by the NDH in Selection Mode. If text is highlighted, vertical dragging of the NDH modifies its font size (Figure 6.9 left), while horizontal dragging allows switches to be toggled to change its font style (bold, italic or normal). This overloads functionality already provided through the popup toolbar but allows rapid phrasing of the most common text-styling operations through smooth sequences of NDH-DH chunks. If the whole text element is selected, or any type of element for that matter, dragging up or down changes the element's z-index to make it appear above or under overlapping objects (Figure 6.9 right). Here as well, gesture relaxation is supported in order to allow single-finger dragging after the four-finger posture has been registered.

As shown in the figure, in both cases the overlay reacts with appropriate visuals displaying how those properties are affected by directional dragging. This method of changing the font size or an element's z-order via a continuous physical action more closely follows RBI philosophy (here NP and BAS) compared to selecting a number or a command from a drop-down menu, for which it is difficult to picture how it translates visually on the sheet (and even more so in vector-based editors). The WIMP equivalent of this technique would be a slider.

6.2.4 Shape and Freeform Input

In Shape Mode, the user can sketch freely on a page. The underlying shape beautifier attempts to detect geometrical shapes and if it recognises any, it smoothes the strokes accordingly. As mentioned above, if a grid pattern is detected a table is created instead of a shape object. The current prototype only features basic table-specific interactions: adding rows and columns through transversal lines and inserting text in cells.

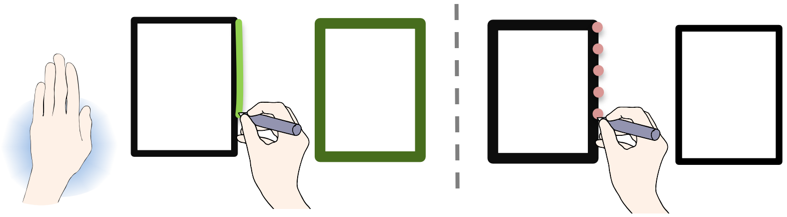


Figure 6.10: Line or curve tracing along shape contour in Shape Mode to increase its width and blend colours (left) or to decrease it if the transparent colour is selected (right)

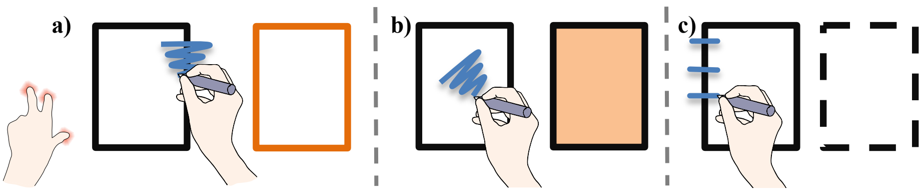


Figure 6.11: Shape styling gestures in Command Mode: scratch gesture to change a) the colour of contours and b) inner areas; c) three short cuts toggle between dashed and continuous lines.

User strokes that "hug" the outline of existing elements are treated differently. For instance, if the user draws a rectangle that completely surrounds an element and that element has no visible contour, a border is created and attached to it. For shape elements and items with borders, curves drawn along their contours with a hard colour cause their thickness to increase and the border colour to be blended with the selected colour (Figure 6.10), in analogy with physical drawing and painting effects. If, however, the transparent colour is active, the thickness is decreased, as if the shape outline was being trimmed by an eraser pen.

In Command Mode, the colours of shape elements can be changed using a scratch gesture, mimicking pencil-drawing techniques to add shading to line sketches. Both fill and contour colours can be modified in this way. Depending on whether the scratch gesture is executed inside the shape or on the contour, the fill or contour colour is replaced by the currently selected colour (Figure 6.11a and b). If performed on free unoccupied space of a page, the background colour of that page is changed.

Also in Command Mode, three short cutting strokes across the contour of a shape outline or element border toggles between dashed and continuous line styles (Figure 6.11c). This DH gesture is similar to that described in [75], but additionally, the average distance between the intersecting strokes is used to determine the dash pattern.

Finally, in Selection Mode, the pen functions as an eyedropper tool to sample colours from objects.

6.2.5 General Commands

Here further operations executed by pen gestures in Command Mode are described:

New Page/Document Creation

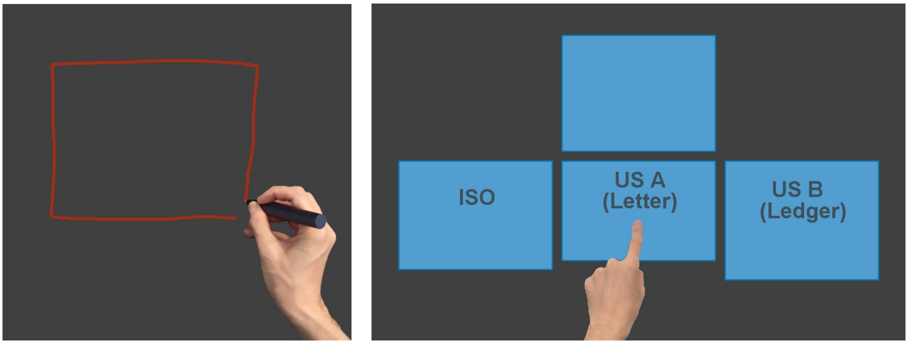


Figure 6.12: Drawing a rectangle to create a new page/document (left). A choice of standard page formats to select from appears (right).

A rectangle shape drawn on the workspace adds a new page if the shape is drawn next to the end of an expanded document or triggers the creation of a new document if drawn elsewhere. When the user completes the gesture, an overlay showing different page formats appears with standard aspect ratios: ISO, US A and B (Figure 6.12). Depending on the length/width ratio of the rectangle, those page samples appear in portrait or landscape format. The user can then select one of the standard formats or the initially drawn rectangle.

Deletion

Line strokes delete elements (pages, document components, text) crossed by them (Figure 6.2 right). If the delete stroke is executed inside a text element, the section of text between the start and end of the line is erased, unless a portion of the text is already highlighted, in which case the strike-out gesture deletes the entire selection.

This gesture is distinguished from the contour dash toggle gesture, which is also based on line strokes, by its trigger area: for the former, the line needs to entirely strike through two edges of its bounding box, whereas for the latter each line must intersect only one segment of the shape contour. As the bounding box has a minimum size requirement, the special case where the shape is a straight line is also adequately taken care of.

6.2.6 Content Registration and Insertion

To meet the requirement of seamless integration of external document data, a special mode dedicated to gesture-based content insertion was included. This mode allows users to associate gestures with specific document elements and recall them by re-executing that same gesture.

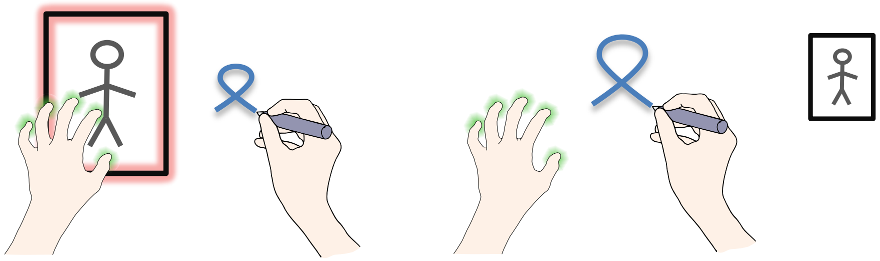


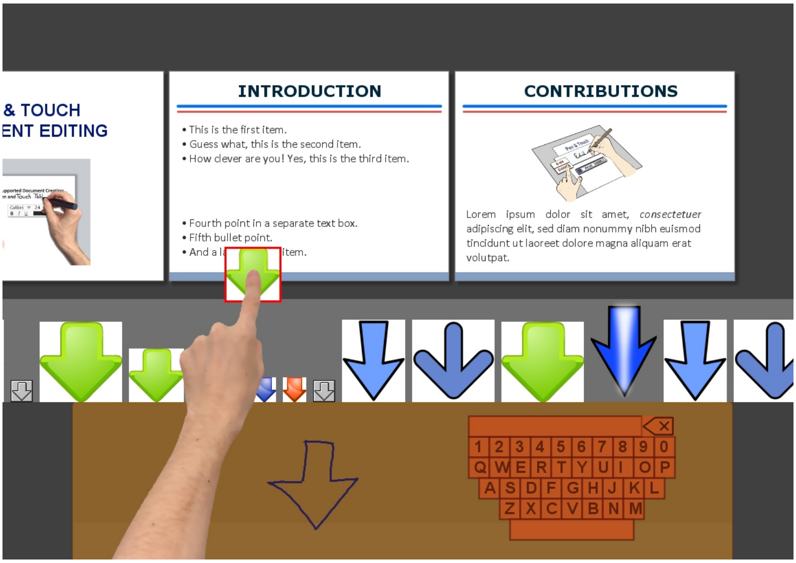
Figure 6.13. Associating and recalling a document element with a user-defined gesture. Content-gesture association (left) and insertion (right).

To register a gesture, the user first selects an element with a single finger of the NDH, lays down the four other fingers to activate Content Mode (which is invoked as a local mode here, as an element is selected) and enters a single-stroke pen gesture to complete the association (Figure 6.13 left). If a gesture is deemed similar to that of a previously registered element, the link is replaced and the gesture is assigned to the new element. To insert a "saved" element, the user simply executes its trigger-gesture on a document page, where the size of the inserted element is computed based on the bounding box of the pen stroke (Figure 6.13 right).

Other than user-specified content gestures, the document editor can conceivably also include pre-defined system shortcuts provided by gesture libraries, for instance often used logos, addresses, parts of text etc.

In many ways, this feature is similar to Quick Parts in Microsoft Word, which allows users to store and find reusable pieces of content via the gallery [13], but instead of creating and recalling those elements using several keyboard and mice operations, all manipulations here are performed using quick pen and touch gestures.

6.2.7 Clipart Retrieval with Query-by-Sketching



INTRODUCTION

- This is the first item.
- Guess what, this is the second item.
- How clever are you! Yes, this is the third item.

- Fourth point in a separate text box.
- Fifth bullet point.
- And a l item.

CONTRIBUTIONS

Lorem ipsum dolor sit amet, *consectetuer* adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

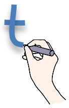
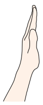
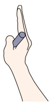
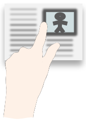
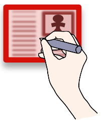
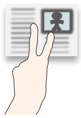
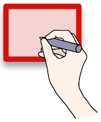
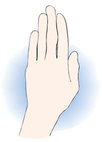
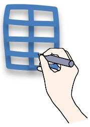
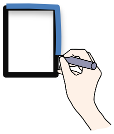
Figure 6.14. A user drags a selected clipart retrieved using a sketch-based query.

In addition to gesture-based content retrieval, the system includes a means to retrieve document elements via queries defined by sketches, which are sometimes more convenient to describe visual items with characteristic geometry (e.g. arrows, chart items, block shapes).

This feature is materialised by a virtual "drawer" panel that can be pulled out from the bottom bezel of the interface. On this panel users can then draw sketches of objects to perform queries using query-by-sketching (in this prototype, the system described in [27] is used) as well as enter search terms if keyword-matching is also desired. For instance, if a user would like to search for an arrow clipart in the database, they can sketch an arrow shape on the pad and additionally (or instead) enter "arrow" as a keyword to further specify the query (Figure 6.14). Keywords are input using a compact virtual keyboard, which is more straightforward for that purpose. It should be noted that keywords are command parameters and not inked document content, which means they do not have to conform to the pen-on-paper metaphor and thus be handwritten in.

Search results are displayed in a scrollable ribbon that appears above the query pad. Those results are updated after each pen stroke or key input. The user can then drag desired elements from the ribbon to insert them in opened documents on the workspace. This method to retrieve and insert external content is relatively straightforward and efficient, but one could also consider alternatives, where query sketches are directly drawn at the desired insertion location on the target document page (as with the gesture-based content insertion technique) and results appear in place or next to the query.

6.2.8 Function/Gesture Summary

NDH	DH	Function
		Text input and underlining
		Text alignment
		Copy element
		Create new blank page of same format
		Shape and table input
		Increase/decrease contour stroke thickness – add border

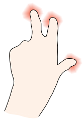
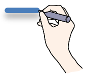
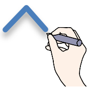
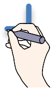
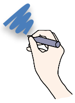
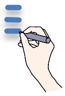
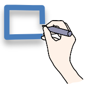
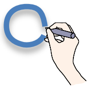
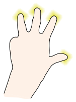
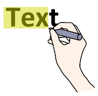
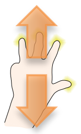
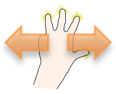
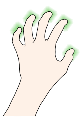
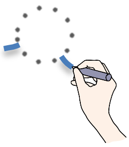
		Delete element
		Insert space
		Insert line break
		Fill with colour
		Toggle dashed stroke
		Create new page with custom format
		Undo/redo
		Element selection/text highlighting
		Change font size – change element z-order
		Change font style
		Register/insert document element

Table 6-1: document-editing functions performed with gestures listed by NDH posture

Table 6-1 summarises the main editing and command interactions of the current system listed according to NDH postures and modes, when applicable.

6.3 Software Implementation

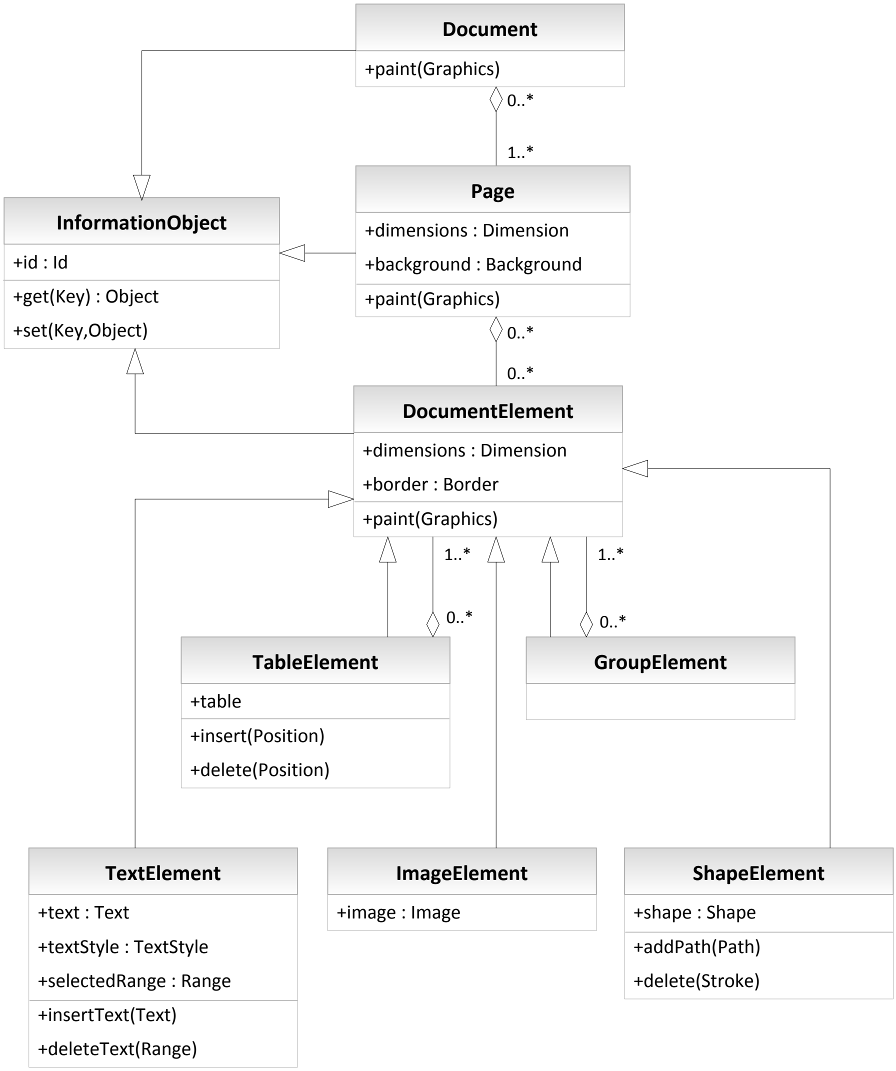


Figure 6.15: Summarised document model

As with the active reading prototype, the pen and touch framework formed the backbone of the application and its components. Because this time, the program involved a real editor dealing with multiple documents containing a variety of elements, a proper document model was required to provide adequate support for the management and rendering of the underlying data.

6.3.1 Document Model

The document model reflects the type of document supported in the application, i.e. an ordered collection of elements with absolute positions on a series of pages. For the needs of the prototype, a simple 3-tier tree structure *document–pages–elements* is sufficient to encapsulate content and layout information (Figure 6.15). Elements include common types of media and containers: text, image, shape, table and group, with appropriate methods to access and set their attributes.

Document objects are further backed by a transparent storage mechanism running in the background, with each element extending an `InformationObject` through which state updates are processed. This connector object is responsible for storing its subclassing entity with its properties. This is achieved simply by an internal hashtable containing descriptive keys and their associated values (`get set` methods). The hashtable is maintained by the underlying storage manager through a `History` object (not represented) that tracks changes made to the data stored in the table. The `History`, as its name implies, keeps a record of all transactions, which is used, among other things, for the undo/redo operations.

To facilitate element reuse and sharing, each child element is stored independently of its parent. Hence, an image or a text frame is not aware of the page it is contained in (if any), and pages, in turn, do not hold any references to their container, the root document. This system effectively amounts to a dynamic database of document elements, that, if the required metadata is present, can be searched and retrieved to be inserted in new compositions. Because new user-created elements are also automatically stored, the database is continuously enriched with personalised content that is always quite literally at the user's fingertips. This feature is utilised in the two document element retrieval functions of the editor, i.e. clipart searching using query-by-sketching and element insertion with stroke gestures.

6.3.2 UI and Interactions

Similar to the active reading application, the UI components containing documents and their elements are extensions of `InteractableComponent`, where for rendering, the pseudo 3D appearance of documents with page-turning animations is replaced by a flat page layout with individual subcomponents representing each page.

As evident from the description of the UI above, there is a much more systematic reliance on posture-based modes and pen gestures. Those modes, postures and gestures are implemented as described in Chapter 3. Hand-drawn or -written content is again treated as stroke input until the required criterion for conversion into vector shapes or typeset text is met and an appropriate gesture event is fired. Here, because the default pen mode is freeform text entry, i.e. there is no validation action to indicate that input is complete, the criterion to trigger the conversion is simply a short timeout after the last pen released event. This scheme, where immediate and continuous feedback on pen input can be combined with delayed conversion of the strokes through event notification, is another example of how the framework constructs can be effectively applied to react to the same user input but with different synchronicity requirements. The framework and the document model allow to conveniently separate the implementation of the relevant aspects of the UI and tidily tie them where needed. Hence, input event processing and rendering logic can be clearly structured, based on the currently active mode and gestures differentiated by type and target UI elements.

Finally, the content transfer module is used to manage the various document element moving and copying operations that occur between pages, documents and widgets such as the clipart ribbon.

6.4 Lab Study

6.4.1 Goals

To assess the pertinence of the approach, an informal lab study with 12 participants recruited among students and staff of the university (5 females and 7 males with one left-handed user) was conducted. All testers were familiar with touch interfaces thanks to smartphones or tablets they owned. They were however not used to operating a digital tabletop with simultaneous bimanual pen and touch input. Participants all knew how to use office software as part of their regular work, especially word processors and presentation creation applications, and so were able to formulate precise requirements and criticism regarding specific features of the system based on their experience.

Considering the breadth of interactions that needed to be demonstrated within the limited time of the study, the goal was mainly to observe users and obtain qualitative feedback on the individual gestures after introducing them one after the other. When describing the rationale behind the NDH modifier postures, the keyboard shortcut analogy was explicitly mentioned, in order to find out to what extent users related the two concepts.

Participants were not expected to remember all of the techniques (much like people most likely would not memorise a plethora of keyboard shortcuts presented all at once) and hence they were just asked to comment about the appropriateness of the gestures to perform a specific action, or informally "if they made sense" in the given context. After showing the different interactions, participants were given the chance to use the application freely and attempt to author their own documents using the different techniques they had been taught. General impressions about the application were finally gathered and suggestions about improvements recorded to possibly include in future iterations of the system.

A session with a participant lasted about an hour.

6.4.2 Results

Generally, users gave very positive feedback about the system with many participants staying for extended periods of time to "play around" with the application. The overall feeling was that the gesture-function mappings were fitting and the majority of users, if not all of them, were confident that the different interactions could be mastered in a short amount of time. Popular features included the content retrieval techniques (both query-by-sketching and gesture-based insertion), the chopping interactions for text alignment and the circle-based undo/redo gesture. Regarding the parallel with keyboard shortcuts, several users commented that gestures were easier to remember since the mental association between a particular operation and a physical hand or pen action is a more effective mnemonic than a combination of keys. Those statements confirm previous results about the superior memorability of stroke-based vs. keyboard shortcuts [29]. In the case of the tabletop application, even, the benefits might be greater considering many of the editing gestures are natural expressions of the functions they are associated with. Compared to purely symbolic mappings of transactions to random stroke gestures, concentric circles for undo/redo or drawing a rectangle to create a new page etc. carry a strong psychomotor relation between the physical motion and the produced effect (another RBI principle) and so will probably stick to memory faster.

Following is a summary of the main observations and participants' reactions with regard to some of the more salient features and aspects of the system.

Asymmetric Bimanual Interaction

A general observation that was made and that echoes the experience of other researchers [193, 213] is that users are heavily influenced by their interactional habits with devices they often use, especially with respect to unimanual/bimanual input. The vast majority of applications available for commercial touch devices are based either on single-handed interactions or symmetrical two-handed input, e.g. for typing and pinch/spread scaling. Hence, people are not used to asymmetric combinations involving two different modalities and are therefore sometimes hesitant to engage with this new kind of interaction, even though productivity gains can be achieved. For instance, at times, users temporarily transferred the pen from their DH to their NDH in order to perform a unimanual touch interaction and returned the pen to their DH after they had finished. For most non-inking operations, however, the system forces users to use both hands simultaneously and while participants quickly came to grips with the modifier posture model, whose sequences of actions are consistent across all different modes, some people were confused by locally modified pen gestures, especially those governing page-copying manipulations. They wondered why they had to bother placing a certain number of fingers on pages to move or duplicate them instead of simply selecting the elements first and then executing the desired operation, as determined by a global modifier posture or a button selection. One user also deemed that because creating a new blank page was a more common action for him compared to copying the entire page with its content, the operation should have been assigned to the single-finger hold posture. This would however have broken the consistency with the duplicating interactions for other elements, which are based on the same gestural pattern.

Text Input and Styling

Participants did not experience major difficulties or express concerns about the text entry techniques, which is likely due to the quality of the MyScript handwriting recogniser and the fact that most of the interactions mirror established behaviours of word processors and text-editing software. Even the somewhat unconventional method to insert handwritten content in the middle of typeset text did not cause particular inconveniences (the left-handed participant was particularly at ease, as in his case there was no occlusion of the shifting right portion of text), other than the occasional misrecognitions of insertion points and text that was not properly appended to its neighbouring element. A few rare problems were registered, where underlining actions that were not perfectly performed below the text resulted in hyphens being added in the text instead.

As for the other text-styling options, users found it convenient to be able to summon the pop-up toolbar with touch and make individual selections with the pen. Especially when there was a need to set several different font options in a row, this could be quickly achieved by sliding an NDH finger on the styling categories to successively activate their submenus and then simply tapping with the pen on the desired item to select it. The NDH-triggered overlay elicited more mixed reactions, with four people explicitly criticising the cumbersomeness of simultaneously having two dragging directions to change two different types of parameters. This view was not shared by everybody, however, as some participants, two in particular, proved very dexterous at controlling text size and switching font styles with lateral taps and sliding actions.

Shape Input and Styling

Shape input also did not present any significant problems. Users were comfortable laying a flat NDH on the surface while sketching and drawing shapes. Switching modes to perform styling operations was also not an issue and the only mistakes were a few instances in which some people tried to thicken/thin shape contours in Command Mode instead of doing so in the regular Shape Mode.

Gesture Relaxation and Crosstalk

The modifier posture model was designed to favour efficient mode switching with minimal mental and physical effort of the NDH and so the posture was not required to be completely released in order to change to another mode. This meant that, for instance, a colour sampling action in selection mode could be immediately followed by a filling operation in Command Mode by simply lifting a finger. But while participants appreciated this flexibility, they were confused by mode changes occurring when relaxing dragging gestures (i.e. when lifting fingers) initiated in Selection Mode. In particular, border colour changes associated with mode switches distracted many users while the styling overlay remained visible throughout dragging actions.

Thus, trying to optimise for mode-switching speed and the ability to perform smooth, effortless transitions may sometimes clash with relaxation policies and location-based gestures. In such cases more conservative gesture design approaches might be in order to appeal to a greater number of users.

Suggestions for Improvements

A few participants made suggestions on how some of the features of the system could be improved or extended. The following are the most interesting ideas, some of which have been addressed in other works:

- Selection of occluded elements: when elements completely overlap on a page, it is difficult to select one of the hidden objects for manipulation. Since this issue was not particularly dealt with in the prototype, one participant suggested that Selection Mode could also be used with the DH (without the pen) to "drill down" into the pile by repeated tapping. Specifically, elements from the topmost to the lowest would be selected one after the other with each tap, with dismissed elements turning semi-transparent to allow the lower elements to become visible. Because this technique could only be activated in Selection Mode, it would not override the default behaviour of "tapping selects the topmost object" in normal mode that most users would expect. Furthermore, contrary to other methods such as Tumble and Splat [158], it would not modify the layout of the pile, so element-positioning consistency would be maintained.
- Incremental undo/redo with timeline of command history: the circular pen gesture in Command Mode triggers an undo or redo with each closed circle, regardless of its dimensions. One user proposed that the number of undone or redone operations could be proportional to the size of the drawn circles. Realistically, there would be only 2 or 3 multipliers corresponding to specific threshold circle sizes (as with multi-speed scrubbing techniques for media players [8]) so that users would have precise control over how many undo/redo actions they execute at each step. The participant further suggested to show a timeline with representations of commands occurring before and after the current operation, with a possibility for the user to directly select a state to return or to advance to. While potentially of use, the inclusion of such a timeline would also increase the load on the interface, which might confuse users more than boost their productivity.
- Element alignment and distribution: one user remarked that the chopping gesture could also perhaps be used to align not only text, but also arrays of elements. Chops would be bidirectional and trigger horizontal and vertical alignment depending on the angle. As a matter of fact, chopping as a general alignment tool was considered during the gesture design phase. The problem is that, in most cases, the hand's edge lacks the necessary precision, as it is too coarse to unambiguously determine which elements should be part of the alignment operation. Aligning all elements on the line extending the chop would also not be an option, as the user may not want to include them all. A useful set of techniques for element-aligning on tabletop interfaces are Frisch et al.'s NEAT [77].

Those interactions could conceivably be adapted to be integrated into the gesture model of the application, perhaps with a dedicated mode.

– Integration of web content: two participants expressed they would find it useful to have a web browser available within the application to copy and reuse content from web pages in their own documents. This option was also considered in the requirement definition phase, but in terms of external content insertion for the first prototype, priority was given to clipart retrieval and gesture-based element recall, as some kind of web browser had already been included in the active reading application, albeit without the possibility to extract data. Zeleznik et al. show an example of how web content can be clipped out and inserted in user documents using pen and touch [213]. A similar technique could easily be integrated in the editor. A web browser could, for example, be summoned by a 'W' pen gesture in Command Mode.

6.4.3 Discussion

The experiment reveals that many common document operations can be successfully executed using appropriate unimodal and bimodal gestures. The feedback given by the study participants confirms that modifier postures associated with pen gestures are potent equivalents to keyboard shortcuts and that they are likely quicker to acquire than the latter [29]. In this implementation, gestures were directly shown to users to try out individually as a first test. The problem of discoverability or learnability was deliberately not addressed at this stage, as the focus was more on gesture design and execution. The issue of gesture learnability has been tackled by other authors in dedicated works [35, 42-44, 74]. A possible idea that would fit in the current design would be to include demonstrative illustrations of available gestures for each mode within the gradient border instead of simply displaying a uniform shade. That way, users could simply cycle through the modes to find out which gestures exist.

Regardless of accessibility and ease of use, one must acknowledge that gestures require some initial effort that not all users are necessarily willing to invest. The goal here was not to show that gesture-based interactions can entirely replace widgets and other traditional tools (especially for a feature-rich application such as a document creator), rather that they can be a useful additional asset for experts, much like keyboard shortcuts complement GUI controls of PC applications rather than replace them. Besides, as people are increasingly exposed to direct input interfaces (with increasing use of simultaneous pen and touch) and those interfaces themselves continue to evolve, users will likely become more familiar with hybrid interaction techniques, and within that context, bimodal gestural shortcuts stand a reasonable chance of gaining wider adoption. For that to happen, UI designers will have to create coherent and consistent gesture sets that neither overwhelm nor frustrate users. Compound interactions, in particular, should not consist of too many chunks resulting in exceedingly long phrases [54], especially if they require a great deal of bimanual coordination. A sequence NDH-posture→DH/pen-gesture→ bidirectional NDH-posture dragging as with the text-styling technique is perhaps already too complex.

With regard to in situ handwritten text input and conversion, the technique used in this prototype naturally finds its place in such pen-based environments, where a smooth flow and seamless interleaving with other stylus interactions are desirable. While this would need to be formally confirmed by an appropriate test, one can hypothesise that the throughput loss incurred by handwriting text compared to keyboard typing is compensated by productivity increases in other stylus manipulations, whose performance would otherwise be impacted by repeated context switches with the keyboard and it being "in the way". The document-authoring scenario is one example, where arguably such an approach is justified (albeit, for particular categories of documents) but there are doubtless many others.

6.5 Conclusion

In this chapter, a multi-document-authoring environment for interactive tabletops was presented, where several common document-editing operations are supported by gestures instead of, or in addition to, widgets. Those gestures include a range of unimodal gestures used mainly for navigation and basic content manipulation as well as several hybrid interactions, based on local or global modification of pen modes using NDH postures. This combination of touch-modified postures with pen stroke gestures to form bimanual commands was likened to keyboard shortcuts on a desktop computer. Along with interactions to create and manipulate user input, two types of integrated functions to seamlessly retrieve and insert external content such as cliparts were also described. Finally, a lab study confirmed the potential of this approach and provided insights about the practical challenges of its realisation.

From here on there are many avenues that could be pursued, considering the breadth of functionality typically available in professional publishing software. Some extension possibilities were suggested by study participants, although the realisation of many of those ideas would merely consist in integrating techniques of other works. In terms of novel challenges, of particular interest are table manipulation and templates, which would likely also greatly benefit from a multimodal gesture approach. Tables are essential components of office documents such as spreadsheets and forms and there is much to explore in that area. As for document templates, they can function as constraint structures that can help solve some of the ambiguities, which the application tries to resolve through algorithmic inference. For example, a constraint-defining template would allow text input in a title box to be automatically converted to the correct font and size.

With regard to evaluation, task-based experiments will be required to determine if users are able to effectively learn and utilise the different gestures to create or edit entire documents on their own and how efficiently they can do so compared to traditional document-authoring software on desktops. Of course, ultimately, after the system has become sufficiently robust and mature, it would be most interesting to deploy it in a real office and perform a field study with actual users.

7 Conclusions

In this thesis, the argument was made that "*post-WIMP interaction and UI designs taking advantage of the affordances of digital tabletops and hybrid pen and touch input allow the creation of intuitive and efficient applications for productivity document engineering work such as document editing and authoring*". The goal was to lay the groundwork for future tabletop publishing systems based on this interaction paradigm, approaching the problem from software engineering and HCI perspectives. From the design of a dedicated pen and touch software framework to the implementation of a document authoring application prototype, this effort expounded the various challenges of producing complex but effective tools, while coping with particularities and limitations of the underlying hardware. The general upshot is that a pen and touch tabletop environment can indeed be very effective for certain kinds of productivity document tasks (and certain kinds of documents), if the interaction paradigm is appropriately leveraged.

This chapter concludes the dissertation with a summary of the presented work and contributions, followed by a discussion of salient issues pertaining to the topic, before offering a vision for future tabletop-based document engineering systems and applications.

7.1 Summary

7.1.1 Review

After reviewing prior work and presenting the pen and touch input paradigm in Chapter 2, Chapter 3 introduced a software framework designed for pen and touch interaction on digital tabletops, with particularly the DiamondTouch and Anoto pens in mind. The chapter laid out the touch input ambiguity problem of the DiamondTouch and pointed out the impossibility of abstracting input data for such hardware based only on touch contact shapes (usually ovals), which most existing frameworks relying on the TUIO 1.1 or similar protocols take for granted. It justified the need for a dedicated software framework to support the development of pen and touch applications with special consideration for pen mode-switching triggered by the NDH. The proposed architecture is divided in five layers representing the different levels of concerns. At the lower levels dealing with the encapsulation of sensing data, the Touch Adapter offers a solution to the aforementioned abstraction problem by defining the global bounding box and chords (i.e. number of fingers in contact with the surface) as common work data for the upper layers. The management entities of those upper layers include a Mode Connector, through which physical hand postures can be associated with specific modes, a Gesture Manager, for the definition and integration of pen and touch gestures and a Component Manager, responsible for the administration of UI components. Finally, the topmost Component Layer provides a hierarchy of extensible UI components with different levels of integrated behaviours following a set of standard pen and touch interaction patterns, as well as a Content Transfer mechanism to facilitate exchange of content between components. All the major entities were presented in context using UML diagrams and code samples were provided to illustrate how those software constructs could be utilised in practice.

The following chapter explored various properties of pen and touch interaction by means of four controlled experiments. The first trial, an object-docking task, compared different moving and scaling methods based on classic multitouch interactions and edge control handles operated by touch or the pen. Contrary to other docking tasks in the literature, the test protocol required subjects to entirely release the object in order to complete the docking action. This allowed the measurement of a displacement on release effect, i.e. the slight shift of the contact shape upon finger or pointer release, which often causes the held object to move out of position. The experimental displacement values obtained for single-finger touch and the pen were respectively 1.2mm and 0.6mm, which provides an indication of how much smoothing or thresholding is required to filter out this effect. Other observations showed that pinch/spread scaling is not adequate for precise positioning and that a strategy based on sequencing transformation operations, as made possible by the control handles, is preferable.

The second test considered a tracing task, where the accuracy of direct finger and pen tracing were investigated, the latter with and without allowing palm resting. The main result of this experiment is that palm resting essentially affects the precision of initial targeting and pointing more than overall tracing accuracy. A consequence of that finding is that palm resting is likely less critical for tracing tasks involving long strokes compared to, say, art sketching, where typically many small strokes are drawn in succession. Of course, this is only valid for relatively short pen tasks, as otherwise arm fatigue becomes an issue as well. The other lesson from this experiment is that pen and touch have comparable precision in rapid tracing situations, i.e. when the temporal demand is high.

The third experiment tackled the issue of NDH-driven pen mode switching between inking and commands in the context of a simple shape-drawing task. The three tested mode-changing techniques were a popup radial menu and a toolbar with and without maintaining of non-inking modes. The results did not show any significant differences between the three configurations in terms of efficiency and user preference. The experiment also confirmed observations by other authors that users would also like to use the pen for basic widget manipulations, such as menu-selection and button-tapping. Designers of pen and touch interfaces should therefore not enforce a strict division of labour between pen and touch for inking and non-inking tasks.

The fourth examination delved deeper into the question of quasimodal mode-switching by the NDH with a task requiring further coordination effort between the two hands. The assignment required participants to trace with a single stroke over a polygon consisting of segments with different styles (plain, thick, dotted, dashed), which had to be matched on-the-fly using NDH-triggered switching actions. Here again, three methods to select the style of the tracing stroke were studied: normal toolbar selection and NDH postures, with and without the constraint that they should be maintained (i.e. with and without quasimode). The outcome of this experiment was that users were indeed faster switching with maintained postures, with also less coordination effort required, but at the price of increased mode errors. Observations led to suppose, however, that those mistakes could decrease with sufficient practice. Another point that was highlighted in this experiment was the need for appropriate awareness mechanisms when such type of blind mode-switching strategies are utilised.

Chapter 5 introduced the first pen and touch application in a document scenario of moderate scope: active reading. The prototype, whose interface is based on a loose codex metaphor, features a number of tools to navigate, mark and search within documents in a manner that is close to or partially inspired by real pen on paper interaction. Thus, in addition to single-page turning, a gesture to mimic leafing through multiple pages is included. Furthermore, annotation and search results concerning non visible items are materialised as dynamic page markers and bookmarks, which can be tapped to flip to the corresponding page. Switching to non-inking modes for the pen is achieved by activating and maintaining a finger on a function-determining hotspot area and entering the parameters of that function with the pen. Commands implemented in that way are deleting, turning to a specific page and searching based on handwritten or hand-drawn input. Finally, a chop-and-spread bimanual multitouch gesture is available to trigger an overview of all the document pages.

A user evaluation was conducted to assess how far the system was able to combine the benefits of paper and computers but without most of their disadvantages. The study consisted of three active reading tasks with different purposes: locating visual elements, searching for answers in text and basic editing. Those tasks were performed on three platforms: paper, a desktop PC with Adobe Acrobat and the tabletop. The study revealed a number of positive results. First of all, in terms of efficiency, the tabletop application systematically tied at first place with the other best-performing medium, which was either paper or Acrobat. The users further rated their experience on the tabletop higher than on the two other platforms. The usefulness of most of the functions of the system was also acknowledged by the majority of users, especially the document overview and the search functions. On the negative side, some interactions such as the dragged page turns (single and multiple) were met with more scepticism. The success of gestures like the chop-and-spread overview, which departs from the paper metaphor, compared to the failures of others that were modelled after physical paper interactions suggest that interface designers should not try to recreate or mimic real-world conditions at all costs. Finally, a survey about the tabletop application's hardware and implementation limitations revealed that the greatest handicaps in the eyes of users were text readability and pen lag issues. Although a large part of those problems most likely can be attributed to Anoto's technology (specifically the dot pattern and the slowness of the pens), those results give a hint at what types of criteria are important for users when engaging with documents on a digital tabletop system. In the last part of the chapter, a few ideas to improve the system and to address further requirements of active reading were given.

The second document scenario considered a full document-authoring system with more extensive reliance on pen and touch gestures. The goal for this effort was to create an environment for users to author rich electronic documents using a combination of natural sketching techniques and gestural commands. As a background to the application design approach, a formal model for the construction of bimanual and bimodal interactions from atomic actions was described, with a systematisation of NDH-controlled pen modes. The distinction was made between global modes triggered anywhere on the interactive surface and local modes activated on a particular element, which can be used as an additional parameter for objects pointed at by the pen. Within this framework, the concept of modifier postures was introduced, establishing an analogy with touch-modified non-inking pen commands and keyboard shortcuts. An elementary document model was also presented with the characteristic that document elements are stored independently in order to facilitate reuse in multiple documents. The interface and its interactions were then described, with a design philosophy less bound to the paper metaphor (although not totally abandoning it) and instead drawing inspiration from reality-based interaction [99]. Thus, the virtual codex and the drag page-turning gestures, for which testers of the active reading application showed little enthusiasm, were abandoned in favour of a design based on expandable piles and linear page layouts. A set of multitouch gestures was implemented for navigation and basic page and element manipulations, where users are able to move and scale pages even while touching elements contained in them, thanks to conflict-avoiding chord requirements upon gesture registration. For page duplication, the finger hold + pen drag gesture used in other work is extended to support the copying of ranges and the creation of blank copies. Text entry is handled in a direct way: phrases are handwritten and converted to typeset text in situ, with a possibility to insert handwritten words directly in the middle of converted text elements. Furthermore, a novel use of hand-edge gestures allows users to set horizontal text alignment with single and double chops. The application numbers five global modes, in which pen stroke gestures perform different categories of document operations. Examples of such interactions are rectangles for page creation, scratches, cuts and trace-overs for shape and border styling and repeated circles for undo and redo. Finally, two methods of recalling and inserting external content were presented: the first one involves associating document elements with user-defined stroke gestures and the second is the integration of query-by-sketching for clipart retrieval.

The lab study evaluating the pertinence of the interface and the gestures gave positive results, with users expressing agreement with most of the design choices. The keyboard shortcut analogy resonated with a majority of the participants and observations (supported by prior research results [29]) led to believe that the different gestures could be relatively quickly mastered given a minimum of practice. The main issues raised by participants concerned posture dragging motions to carry out font styling actions, which they viewed as too complex and error prone, and chord gesture relaxation resulting in mode changes causing distracting visuals to appear. Although a more formal task-based evaluation would be required for more rigorous validation, those results strongly support the hypothesis that pen and touch digital tabletops are very suitable platforms for efficient document authoring, albeit for certain types of documents only.

7.1.2 Summary of the Main Contributions

Following are the main contributions of this thesis, summarised in a list:

- A software framework for pen and touch application development with the following novel characteristics:
  - A unifying input abstraction layer supporting both tabletop hardware capable of detecting multiple fingers and contact shapes and the DiamondTouch
  - Explicit support for intermodal design, in particular mode-based bimanual interactions and gestures
  - A pen and touch component hierarchy with built-in behaviours for standard manipulations and inking
- A deeper understanding of the properties of the pen and touch input paradigm. In addition to the corroboration of several results reported in prior work the following new findings were established:
  - The displacement-on-release effect measured at 1.22mm for touch and 0.6mm for the pen
  - Palm resting affects targeting accuracy more than tracing precision
  - Quasimodal pen function modifying is quicker and requires less coordination with postures compared to regular widgets, but cause more errors
- A pen and touch-based application for active reading integrating paper-like interactions and augmented with digital tools to navigate and search content. Novel interactions include:
  - A two-finger multiple-page turning technique mimicking real page flicking.
  - Functions activated by maintained selections using the NDH and executed with the pen to search and turn to specific pages through handwritten input anywhere on the surface.
  - A relaxable chop-and-spread gesture to trigger a hierarchical page overview.

  The main insights obtained from the user evaluation are:
  - When compared with paper and a classic desktop PC environment, the tabletop application performs on a par with one or the other medium, depending on the active reading task.
  - The paper/desktop metaphor should not be pushed too far for the sake of recreating interactions of the physical world. Usability concerns call for tools designed with practicality in mind so a delicate balance must be struck.

    - Viewing resolution is a critical factor for document applications on large screens (an aspect that is often underestimated).
- A gesture-based pen and touch application for full document authoring with the following original features:
    - A 5 degree of freedom model to construct bimanual pen and touch interactions
    - A novel in situ text input technique allowing insertions of handwritten content within typeset text and hybrid styling using widgets or pen gestures when appropriate (in particular, text alignment with hand chops).
    - A coherent set of common document editing operations divided in inking and non-inking categories associated with particular non-dominant hand postures and articulated by pen gestures.
    - The possibility to move the NDH in a third chunk (following mode-setting and pen-based selection of the target object) to modify properties such as text size and style as well as the z-index of elements.
    - Seamless recall and insertion of external content via pen gestures and sketch queries (new pen gestures can also be defined by the user to register custom content)

7.2 Discussion

The simultaneous availability of pen and touch input offers rich interaction possibilities that practitioners are still exploring and applying in different contexts. The digital pen, as a precision instrument wielded by the dominant hand, mainly articulates the phrases of the task, which can be direct inking but also operational, as with lasso selections and deletions. Those phrases are supported by (multi)touch interactions performed by the non-dominant hand (NDH), in accordance with the kinematic chain model [82]. At times, however, pen and touch can carry out the same functions, for instance, for widget selection and activation. The choice to use one or the other for such actions is a matter of user convenience and preference. UI designers may choose to impose or nudge towards a division of labour by placing the widgets close to the NDH and additionally requiring that their activation be maintained during the action. This strategy was adopted in the active reading application and worked relatively well for the few functions that were implemented. The problem with widgets, however, is that they take valuable screen space and can potentially be in the way, especially if placed close to the loci of actions to minimise hand movements. A solution to that problem is to use gestures. The second document prototype showed how several command shortcuts could be realised using bimanual combinations of pen and touch input, based on modifying postures of the NDH. But even those techniques have limitations, as they need to be discovered, learned and remembered, although there are several indications that gestural shortcuts are easier to master than keyboard equivalents, especially if the gesture motions are strongly related to the triggered operation (e.g. circular gestures for undo/redo, which suggest rewinding/fast-forwarding). The conclusion of Chapter 6 hinted that a viable pen and touch interface could therefore mirror the novice-expert set of tools found in WIMP software, with explicit widgets and a few elementary interactions for the former category of users, complemented by more advanced gestural techniques for the latter. The large workspace afforded by a tabletop device arguably lends itself to such designs to a much greater extent than mobile devices with their relatively limited screen real estate, especially with respect to surface-wide bimanual manipulations. As with all gesture-based interfaces, however, care should be taken to make sure collisions are avoided or minimised. The division and categorisation of pen gestures between separate modes facilitates conflict-prevention to some degree, as long as the separation is strict. As we have seen, gesture relaxation can lead to confusing situations.

With regard to the pen-on-paper and workdesk metaphors, a stylus-based platform is an incentive to build UIs that mimic real-world interactions. But as has been demonstrated, designers should not aim to recreate every aspect of the physical experience. Not only will the affordances of paper most likely not be matched in the near future, but such a goal is not necessarily desirable in the first place, as paper has several limitations of its own. Moreover, people are becoming more and more familiar with multitouch devices and so interacting with such systems is becoming increasingly natural. The approach to designing an effective tabletop or even tablet interface for document manipulation therefore should be more pragmatic and grounded more in the functionality to achieve than the environment to imitate. Reality-Based Interaction teaches us how skills from the real world can inspire and be applied to interface design when pertinent, with particular trade-offs to be allowed for, when clear practical gains can be obtained (increased expressive power, efficiency, versatility etc.). A good example of that trade-off in the active reading prototype is the clear preference of users for the more practical overview mode triggered with a quick and simple surface-wide hand gesture over the reality-based multiple page turning gesture that involved more precise finger positioning and dragging. In the document-authoring application, the codex metaphor was abandoned in favour of a flatter design and looser analogies: shake gestures to expand and collapse pages, double-tap to trigger a fit-to-screen zoom etc. For content input and shape styling, RBIs with varying degrees of fidelity were adopted when appropriate. For instance, to convert a stroke style to dashed, pen gestures slicing through the stroke were used. However, only three cuts sufficed to do so, as requiring users to make slits on the entire contour would have been overly tedious.

In terms of hardware, the limitations of the chosen technologies and their impact on document tasks have been clearly identified. For the DiamondTouch, the touch coordinate ambiguity problem requires a different software design approach from most other touch screens, where the TUIO protocol (v1.1) is sufficient for most application needs. The framework developed for the purpose of pen and touch prototype development resorted to workaround solutions to implement features such as chord-detection and different shape postures, despite those constraints. As for Anoto, the main inconvenience, as established by the active reading study, is the reduced legibility of fine-printed text due to optical distortions caused by the dot pattern. Of course, those shortcomings are very particular to the two chosen technologies and one may argue that future pen and touch systems, such as Perceptive Pixel's interactive screens, will not suffer from those weaknesses. Still, no system is perfect and there will always be constraints of some nature to deal with. For instance, the current PixelSense table (Samsung SUR40) is very prone to ambient light interference and incidental contacts. What is more, many of the currently available large touch screens, in particular those based on TV panels, have relatively low dpi values, i.e. low pixel resolutions compared to their dimensions, which is less than ideal for displaying documents. The general lesson that can be derived from this is that it is paramount to know the underlying platform and to anticipate possible effects on application development at an early stage. For document-centric applications, criteria such as viewing and pen-sensing resolutions, responsiveness are particularly important.

A crucial issue that was investigated in terms of its impact on tracing precision but remains to be resolved more generally is palm rejection. Currently there is no reliable method to perfectly differentiate between touches that should trigger a response from the UI and those that should be discarded, especially when bimanual techniques involving shape-based gestures and interactions in which fingers and pen are close (e.g. touch hold + pen drag) need to be supported. This is an area for interesting future work that is, of course, not limited to only document-centric applications.

Finally, another aspect that was not as extensively dealt with in this thesis as it deserves is text entry on interactive surfaces. This is also an active area of research that has currently not produced solutions, whose performance can rival that of physical keyboards on PCs for text-heavy tasks. That is not to say that pens and virtual keyboards are subpar text input methods in all kinds of scenarios. As captured by one of Buxton’s well-known mantras, "everything is best for something and worst for something else" [53]. Therefore, along with inventing or refining means to enter text on digital surfaces, the goal of researchers should also be to identify appropriate matches between those techniques and particular application contexts. As hypothesised at the end of Chapter 6, the stylus is likely a more suitable medium for text input if considered in conjunction with other pen-dependent activities such as sketching and tasks where switching occurs frequently. How far this is true and what are the quantitative performance differences between the different text entry approaches in those different contexts remain to be determined. There are indeed currently very few empirical results available for interactive tabletops in that area.

7.3 Vision

The Office of the Future, by definition is an elusive goal, as it covers an array of visions, anchored in conjectures at a particular point in time about how technological progress will contribute to the evolution of the workplace. As with any long-term predictions, it is difficult to foresee which practices will eventually prevail and what types of infrastructure knowledge workers will have at their disposal. Nevertheless, from Apple's Knowledge Navigator to Microsoft's latest concept videos, one can distinguish a recurrent theme, that is, the reliance on screens and embodied interactions [68], be those voice, pen or haptic. The augmented office desk, despite losing its status as the sole empowering device is still the centrepiece of workers’ electronic armamentarium and interestingly it has been depicted in a remarkably consistent way throughout the years. Thus, both desktop and vertical screen surfaces are fully interactive and operated with audio and gestural commands. As a result of prognostication only marginally aiming further than the previous ones (and also readjusting to envision more attainable short-term goals), technological advances have caught up with the dreams, so that many of them now seem within close reach. Today, multitouch equipment is ubiquitous, including in large form factors. Pen and touch-controlled displays, after a few prototypical experiments in research laboratories, have finally come onto the market and are poised to take the NUI revolution to the office world. There are however still considerable challenges to be met.

Regarding documents more specifically, it is safe to assume that they will still be present in the foreseeable future, as their affordances make them indispensable means of communication. The question is rather in what ways documents and interaction practices will evolve and how future devices will support document work in and outside of the office. So far, multitouch computing has not revolutionised how people carry out so-called productivity tasks. In particular, professional document authoring remains the preserve of PCs with keyboards and mice running WIMP-based software. It will be interesting to see when and how this status quo will break (if it does). Most likely, changes will occur gradually and affect specific domains first, with perhaps some new instruments appearing to complement existing tools to help with the execution of specialised tasks. As hinted at in the introduction, pen and touch tabletops could be used for layout design and drafting, and the keyboard (physical or virtual) to input text content. Another example would be a document-browsing tabletop application in which an expert reviews, classifies and annotates various documents (e.g. for patent review) and sends results back to individuals to make appropriate modifications on their PC. Combinations with mobile devices can also be envisaged, for instance, data gathered on field reports with smartphones and then compiled on a large tabletop surface in the office. There are also no doubt many promising applications to be conceived for individual professions and tasks. Obvious examples are CAD systems for architects and design engineers, but tailor-made tabletop solutions might also exist for accountants, analysts, administrators, civil servants etc.

In terms of document type, it is not yet clear to what extent tabletops can support the whole gamut of office documents. This work showed an example of how an environment for the creation of presentations and DTP-style documents could look like, and while many of the presented concepts can be generalised for other types of documents (particularly touch-modified pen command shortcuts), tailored interfaces will likely have to be created for each category of documents with perhaps an even greater need for bespoke designs than in WIMP applications. One particularly interesting class of office documents that could take advantage of a custom pen and touch application are spreadsheets with integrated chart support. Cell manipulation performed with a single pointing instrument (be that finger, mouse or pen) can conceivably be augmented with the power of bimanual interactions with efficient phrasing of selection and command operations. This also extends to formula input, where target cells can be selected by the NDH and operators handwritten with the pen (and recognised by an ad hoc engine). For charts, works such as SketchInsight [193] have provided a window into the potential of pen and touch platforms to help more directly materialise and interactively visualise such type of information. Those are still the very early stages of tabletop computing for the office and there is still a vast expanse of uncharted waters to explore ahead.

A
Material Used for Chapter 4 Experiments

Pre-Study User Questionnaire

a.	**What is your gender?**		
	1- Male 2- Female		
b.	**What is your age (roughly)?**		
		1	21-25
		2	26-30
		3	31-35
		4	36-40
		5	41-50
		6	51-60
		7	61 or above
c.	**How would you rate your ability to search for information on the Internet using keywords?**		
	1-excellent 2-very good 3-good 4-fair 5-poor		
d.	**How do you search for information contained in documents on your own computer? (circle all that apply)**		
		1	Try to remember location and work from there
		2	Use desktop search
		3	Ask for help

	4	Other. Please specify:
e.	**How familiar are you with Adobe Acrobat's commenting/editing tools?**	
	1	Use them regularly
	2	Have used them once or twice
	3	Know their existence but have never used them
	4	Do not use Acrobat
f.	**Do you own a touch or pen-operated device (Tablet PC, iPhone, iPad etc.)?**	
	no	yes. Please specify which device(s) :
g.	**How often do you use the following methods to input text for processing?**	

Physical Keyboard	1-Very often	2-Often	3-Moderately	4-Rarely	5-Never
Touch Keyboard	1-Very often	2-Often	3-Moderately	4-Rarely	5-Never
Handwriting	1-Very often	2-Often	3-Moderately	4-Rarely	5-Never
Speech	1-Very often	2-Often	3-Moderately	4-Rarely	5-Never

Visual Search Tasks and Material

Set 1

- "Where is the description of the University of St.Gallen?" in *Campus and Research Park Switzerland 2008*, 28 pages
- "What piece of protecting gear is shown on the page introducing research in 'Risk'?" in *Cutting Edge Research Made in Switzerland*, Swiss National Science Foundation 22 pages
- "Where is the photo of the Swiss Sound Box, Swiss Pavilion, Expo 2000?" in *The Pritzker Architecture Prize 2009 Peter Zumthor*, 17 pages
- "Where is the ad for the tour operating company that advertises hotels in Abu Dhabi, Dubai and Oman?" in *Condé Nast Traveller*, Oct. 2006, 14 pages
- "When are the workshops of the conference scheduled?" in Food and Democracy 5th European Conference on GMO-free Regions 2009, 8 pages
- "Where is the information about Nendaz – Veysonnaz?" in *Switzerland Travel Experience*, Switzerland Tourism 2099, 7 pages

Set 2

- "Where is the chart about 'Telecommunications Infrastructure'?" in *Swiss Attractiveness Survey*, Ernst & Young, 2006, 27 pages
- "Where is the section about doctoral research universities?" in *Studying in Switzerland, Universities of Applied Sciences*, Rector's Conference on the Swiss Universities of Applied Sciences, 2010, 22 pages
- "Where are the stamps with birds?" in *A Swiss Sampler, an album showcasing the variety and beauty of Swiss stamps*, American Helvetia Philatelic Society, 20 pages
- "Where does the section about the Swiss University System end?" in *Student Guide to Switzerland*, University of Kent, 11 pages
- "In what street is Schaffhausen's Youth Hostel?" in *Hostelling International Hostels Guide Switzerland*, 8 pages
- "Where is the section about the vegetation of the park?" in *MFO Park, Zurich, Switzerland*, 7 pages

Set 3

- "Where are the recipes?" in Burt Wolf, Travels & Traditions, Zurich, Switzerland, 2001, 6 pages
- "Where is the 'the culinary hear of Switzerland', according to this guide?" in *Swiss Gastronomy 2003-2004*, Switzerland Tourism, 8 pages
- "Where is the section about the Swiss Federal Office for National Topography?" in *Third Annual Switzerland Trip Report 2009*, Schulich School of Engineering, University of Calgary, 17 pages
- "Where is a photo of a Linux washing powder?" in *Visionen 2010-2*, Verein der Informatik Studierenden an der ETH Zürich, 22 pages
- "Where is the information about ice skating?" in *Winter Activities 2009-2010*, Our Chalet, 27 pages
- "Where is the page showing the objects that are fundamentally forbidden to take on a plane?" in *Zurich Airport Guide*, 2007, 17 pages

News Article Compilations for Text Search and Edit Tasks

News Compilation 1 (NC1): Swissinfo

- Cocoa price spike hits chocolate industry, August 16, 2010
- Street Parade ends without major incident, August 15, 2010
- More than a Swiss facelift for Polish hospitals, August 16, 2010
- Federer loses out to Murray in Toronto, August 16, 2010
- Driver faces $1,000,000 speeding fine, August 13, 2010

- Global electric race begins in Geneva, August 16, 2010
- Swiss join in criticism of US entry fee, August 15, 2010
- Luca and Lara named pram king and queen, August 10, 2010
- Troubled tax haven keeps flag flying, August 14, 2010
- More coherent policy needed for Swiss abroad, August 14, 2010
- Trade deal with China comes a step closer, August 14, 2010
- Chinese film takes top prize at Locarno, August 15, 2010
- Campaign launched for Asian "fashion victims", August 13, 2010
- Furka steam railway line reopens to fanfare, August 12, 2010
- Swiss fired up over arrival of mercenary firm, August 11, 2010
- Ghanaians hone bicycle know-how in Switzerland, August 11, 2010
- Estonia's emergency service dials Swiss number, August 9, 2010
- Swiss prices soar over EU neighbours, August 16, 2010

News Compilation 2 (NC2): Swisster

- Expert campaigns for free wireless access in Zurich, August 16, 2010
- Zurich zoo name contest draws interest in lion cubs, August 11, 2010
- Vaud municipalities abandon plastic recycling, August 12, 2010
- Credit Suisse cuts UK jobs while growing India presence, August 13, 2010
- American diva interprets spirituals for St Prex festival, August 12, 2010
- Profitable Nestlé confident about the rest of 2010, August 11, 2010
- Swiss government set for change next month, August 10, 2010
- Swiss financial fraud case draws Madoff parallels, August 10, 2010
- Valais finds novel way to market 'humane' cow fights, August 9, 2010
- Photographers collide with CERN's universe, August 6, 2010
- Rodin sculptures descend on Vevey shopping mall, August 10, 2010
- Swissquote staff produce novel YouTube video, August 5, 2010
- 'We are hiring,' firms tell KOF economic institute, August 9, 2010
- Gulf oil leak weighs heavily on Transocean profits, August 5, 2010
- Record Swatch sales signal strong industry rebound, August 4, 2010
- 'Slow-up' national festival circuit hits Geneva, August 6, 2010
- Vaud residents 'up in arms' over caravan camp, August 3, 2010
- Lucerne Festival star Jonas Kaufmann opens up, August 9, 2010

News Compilation 3 (NC3): Swissinfo & Swisster

- Swiss stand by Middle East approach, August 12, 2010
- Battery-assisted bikes boom in Switzerland, August 10, 2010
- Deportation flights face renewed difficulties, August 15, 2010

- Poll reveals European mindset among Swiss, August 11, 2010
- Swiss firms record solid half-year results, August 16, 2010
- Neuchâtel geography prof touts ‘compact cities’, Swisster, August 4, 2010
- Unions stake their claim for wage hikes next year, Swisster, August 3, 2010
- Locarno's 'secret garden' sparks planning protest, Swisster, August 2, 2010
- Magnitude of Pakistan crisis starts to surface, Swissinfo, August 17 2010
- "Super" speed camera to watch over Swiss roads, Swissinfo, August 17 2010
- Young Boys stun "sluggish" Tottenham, August 23 2010
- 'One way' Swiss animal legend still dupes tourists, Swisster, August 17 2010
- Labour campaign against UI changes targets Dougan, Swisster, August 17 2010
- Nestlé factory in Congo sparks mixed reactions, Swissinfo, August 18 2010
- Travellers told not to import plants, Swissinfo, August 16 2010
- Swiss vandal in Singapore gets two more months, Swissinfo, August 18 2010
- Cricket: Raincoats and a tie seal Power Winterthur’s fate, Swisster, August 17 2010
- Getting together to walk the talk, Swissinfo, August 18 2010
- Switzerland ranks second in "best country" list, Swissinfo, August 18 2010

Text Search Tasks

With NC1

- "How many people were treated by paramedics at the Street Parade?"
- "What is the name of the three-wheeled vehicle developed by 'Velos für Afrika' for disabled people?"
- "What was one of the most popular names given to male babies in Switzerland in the 1990s?"
- "How much is Switzerland paying to revamp Estonia's emergency response system?"
- "In what year was the The Furka Cogwheel Steam Railway Club founded?"

With NC2

- "Where are the offices of Swissquote?"
- "How old will the tenor Jonas Kaufmann be this year?"
- "When did Zurich zoo's policy to give names to animals starting with a different letter each year begin?"
- "How many members does Rinaldo Cajochen's Facebook group 'Gratis WLAN in the whole Zurich city' have?"
- "Who finances Rodin's exhibition in Vevey?"

News NC3

- "How much money did canton Geneva receive from speeding fines in 2009?"
- "What is the name of Nestlé Congo's project to help women sell their products?"
- "How many deportation flights from Switzerland were there in 2008?"
- "What certificate do you need to bring a plant in Switzerland from a foreign country?"
- "How much does it cost to replace an e-bike battery?"

Edit Tasks

With NC1

- Circle all dates on page 15 using the red colour
- Circle all country names on page 6 using the red colour
- Cross out the photo caption of the article "Chinese film takes top prize at Locarno" using the green colour
- Cross out the summary of the article "Ghanaians hone bicycle know-how in Switzerland" using the green colour
- Draw a box around all photographs where NO people appear using any colour
- Draw a vertical line in the margin of the text where you made the cross-outs using any colour
- Draw a vertical line in the margin of the text where you made the circles using any colour

With NC2

- Circle all occurrences of "UBS" on page 4 using the red colour
- Circle all city names on page 14 using the red colour
- Cross out the photo caption of the article "Rodin sculptures descend on Vevey shopping mall" using the green colour
- Cross out the summary of the article "Vaud residents 'up in arms' over caravan camp" using the green colour
- Draw a box around all photographs where you can see a natural landscape using any colour
- Draw a vertical line in the margin of the text where you made the cross-outs using any colour
- Draw a vertical line in the margin of the text where you made the circles using any colour

With NC3

- Circle all percentages above 80 per cent on page 4 using the red colour
- Circle all occurrences of the name "Humbert-Droz" on page 17 using the red colour

- Cross out the photo caption of the article "Magnitude of Pakistan crisis starts to surface" using the green colour
- Cross out the summary of the article "Travellers told not to import plants" using the green colour
- Draw a box around all photographs representing a sport scene using any colour
- Draw a vertical line in the margin of the text where you made the cross-outs using any colour
- Draw a vertical line in the margin of the text where you made the circles using any colour

Post-Study User Questionnaire

How would you rate your overall experience with the 3 different media for the given tasks? (Please tick the box that best represents your opinion.)

	Very good	Good	Average	Poor	Very Poor
a. Paper	☐	☐	☐	☐	☐
b. Adobe Acrobat	☐	☐	☐	☐	☐
c. Digital Tabletop	☐	☐	☐	☐	☐

How do you rate the suitability of the 3 different media to fulfil task 1 (visual search)? (Please tick the box that best represents your opinion.)

	Very good	Good	Average	Poor	Very Poor
d. Paper	☐	☐	☐	☐	☐
e. Adobe Acrobat	☐	☐	☐	☐	☐
f. Digital Tabletop	☐	☐	☐	☐	☐

How do you rate the suitability of the 3 different media to fulfil task 2 (text search)? (Please tick the box that best represents your opinion.)

	Very good	Good	Average	Poor	Very Poor
g. Paper	☐	☐	☐	☐	☐
h. Adobe Acrobat	☐	☐	☐	☐	☐

i. Digital Tabletop	☐	☐	☐	☐	☐

How do you rate the suitability of the 3 different media to fulfil task 3 (document edits)? (Please tick the box that best represents your opinion.)

	Very good	Good	Average	Poor	Very Poor
j. Paper	☐	☐	☐	☐	☐
k. Adobe Acrobat	☐	☐	☐	☐	☐
l. Digital Tabletop	☐	☐	☐	☐	☐

Disregarding the limitations of the prototype you just tested, how useful and convenient do you find the ability to use both touch and pen input to manipulate/edit documents)? (Please circle what best represents your opinion.)

Very Useful	Moderately Useful	Prefer Touch Only	Prefer Pen Only	Both worse than traditional computer environment	No Opinion

Describe how useful you found the following functions of the digital tabletop: (Please tick the box that best represents your opinion.)

	Extremely	Very	Moderately	A little	Not at all
a. Page-turn animations	☐	☐	☐	☐	☐
b. Tap page corner once to turn page	☐	☐	☐	☐	☐
c. Drag page corner to turn page manually	☐	☐	☐	☐	☐
d. Tap corner+drag with other finger to flip through multiple pages	☐	☐	☐	☐	☐
e. Handwrite page number to turn to	☐	☐	☐	☐	☐
f. Handwrite keyword to search in document	☐	☐	☐	☐	☐

g. Highlight searched keywords in document	☐	☐	☐	☐	☐
h. Bookmark tabs for keyword searches with context snippets	☐	☐	☐	☐	☐
i. Bookmarks for user annotations	☐	☐	☐	☐	☐
j. Colours in annotation bookmarks reflecting amount of strokes on referenced page	☐	☐	☐	☐	☐
k. Document overview with page thumbnails to select	☐	☐	☐	☐	☐
l. "Chop and spread" gesture to trigger the thumbnail overview	☐	☐	☐	☐	☐

How severely was your experience on the digital tabletop affected by the following limitations? (Please tick the box that best represents your opinion.*)*

	Extremely	Very	Moderately	A little	Not at all
a. Touching not always detected by the system	☐	☐	☐	☐	☐
b. Working on horizontal surface	☐	☐	☐	☐	☐
c. Surface not large enough	☐	☐	☐	☐	☐
d. Having to write without touching the table surface or having to wear glove to use pen	☐	☐	☐	☐	☐
e. Shadows beneath hands covering projected image	☐	☐	☐	☐	☐
f. Difficulty to read pages (text too small, not sharp enough etc.)	☐	☐	☐	☐	☐
g. Pen lag (strokes don't appear immediately)	☐	☐	☐	☐	☐
h. General slowness of the application (other than lag)	☐	☐	☐	☐	☐
i. Difficulty to perform operations requiring simultaneous touching (multiple page flip, chop gesture for page overview)	☐	☐	☐	☐	☐
j. Difficulty to perform operations requiring simultaneous touch+pen (pen page turn +	☐	☐	☐	☐	☐

	keyword search)					
k.	Misrecognised words and/or page numbers written with pen	☐	☐	☐	☐	☐
l.	Other. Please specify:______________________________	☐	☐	☐	☐	☐

Other comments and remarks

Bibliography

1. http://www.documentengineering.org/

2. http://www.adapx.com

3. http://anoto.com

4. http://www.anoto.com/paper-applications.aspx

5. http://www.apple.com/iwork/

6. http://omgpop.com/drawsomething

7. http://mi-lab.org/products/interactive-surface-kit

8. *iPod Touch User Guide*. Apple Inc., 2011.

9. http://www.microsoft.com/en-us/kinectforwindows/develop/

10. https://developer.leapmotion.com/

11. http://www.livescribe.com

12. http://www.microsoft.com/office/vision/

13. http://office.microsoft.com/en-us/word-help/quick-parts-HA010370568.aspx

14. http://office.microsoft.com/en-us/onenote/

15. http://www.pixelsense.com/

16. http://www.microsoft.com/en-us/news/press/2012/jul12/07-09WPCDay1PR.aspx

17. http://www.n-trig.com/

18. http://www.papershow.com

19. http://www.perceptivepixel.com/products/active-stylus

20. http://www.phatware.com

21. http://msdn.microsoft.com/en-us/library/windows/apps/hh465415.aspx

22. http://www.visionobjects.com/en/myscript/

23. http://www.wacom.com/en/products/cintiq/cintiq24touch.aspx

24. http://www.wacomeng.com/touch/WacomFeelMulti-TouchFAQ.htm#_Toc331140777

25. Accot, J. and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, ACM (1997), 295-302.

26. Agarawala, A. and Balakrishnan, R. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM (2006), 1283-1292.

27. Al Kabary, I. and Schuldt, H. SKETCHify - an Adaptive Prominent Edge Detection Algorithm for Optimized Query-by-Sketch Image Retrieval. In *Proceedings of the 10th International Workshop on Adaptive Multimedia Retrieval*, Springer (2012).

28. Annett, M., Grossman, T., Wigdor, D. and Fitzmaurice, G. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 337-346.

29. Appert, C. and Zhai, S. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 2289-2298.

30. Arai, T., Machii, K., Kuzunuki, S. and Shojima, H. InteractiveDESK: a computer-augmented desk which responds to operations on real objects. In *Conference companion on Human factors in computing systems*, ACM (1995), 141-142.

31. Ashdown, M., Tuddenham, P. and Robinson, P. High-Resolution Interactive Displays. In *Tabletops - Horizontal Interactive Displays*, Springer London (2010), 71-100.

32. Ayres, R. U. and Martinas, K. 120 WPM for very skilled typist. In *On the Reappraisal of Microeconomics: Economic Growth and Change in a Material World*, Edward Elgar Publishing Ltd (2005), 41.

33. Balakrishnan, R., Fitzmaurice, G., Kurtenbach, G. and Buxton, W. Digital tape drawing. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, ACM (1999), 161-169.

34. Ballay, J. M. Designing Workscape: an interdisciplinary experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM (1994), 10-15.

35. Bau, O. and Mackay, W. E. OctoPocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM (2008), 37-46.

36. Belotti, R., Decurtins, C., Norrie, M. C., Signer, B. and Vukelja, L. Experimental platform for mobile information systems. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, ACM (2005), 258-269.

37. Benko, H., Morris, M. R., Brush, A. B. and Wilson, A. D. Insights on interactive tabletops: A survey of researchers and developers. *Technical Report MSR-TR-2009-22*, Microsoft Research, 2009.

38. Bérard, F. and Laurillau, Y. Single user multitouch on the DiamondTouch: from 2 x 1D to 2D. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2009), 1-8.

39. Bi, X., Grossman, T., Matejka, J. and Fitzmaurice, G. Magic desk: bringing multi-touch surfaces into desktop work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 2511-2520.

40. Bi, X., Moscovich, T., Ramos, G., Balakrishnan, R. and Hinckley, K. An exploration of pen rolling for pen-based interaction. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM (2008), 191-200.

41. Bier, E. A., Stone, M. C., Pier, K., Buxton, W. and DeRose, T. D. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM (1993), 73-80.

42. Bragdon, A., Uguray, A., Wigdor, D., Anagnostopoulos, S., Zeleznik, R. and Feman, R. Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. In *ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2010), 39-48.

43. Bragdon, A., Zeleznik, R., Williamson, B., Miller, T. and Joseph J. LaViola, J. GestureBar: improving the approachability of gesture-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 2269-2278.

44. Brandl, P., Forlines, C., Wigdor, D., Haller, M. and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, ACM (2008), 154-161.

45. Brandl, P., Haller, M., Hurnaus, M., Lugmayr, V., Oberngruber, J., Oster, C., Schafleitner, C. and Billinghurst, M. An Adaptable Rear-Projection Screen Using Digital Pens And

Hand Gestures. In *Proceedings of the 17th International Conference on Artificial Reality and Telexistence*, IEEE Computer Society (2007), 49-54.

46. Briet, S. *Qu'est-ce que la documentation?* Éditions documentaires, industrielles et techniques, 1951.

47. Bright, P. Why bother? The sad state of Office 2013 touch support. *Ars Technica*, 2012, http://arstechnica.com/information-technology/2012/07/why-bother-the-sad-state-of-office-2013-touch-support/.

48. Buckland, M. K. Information as thing. *Journal of the American Society for Information Science*, *42*, 5 (1991), 351-360.

49. Buckland, M. K. What is a "digital document"? *Document Numérique*, *2*, 2 (1998), 221-230.

50. Buckland, M. K. What is a "document"? *Journal of the American Society for Information Science*, *48*, 9 (1997), 804-809.

51. Bush, V. As We May Think, *The Atlantic Monthly*, 1945.

52. http://www.billbuxton.com/ActiveDesk.html

53. http://www.billbuxton.com/multitouchOverview.html

54. Buxton, W. Chunking and Phrasing and the Design of Human-Computer Dialogues. In *Proceedings of the IFIP World Computer Congress*, North Holland Publishers (1986), 475-480.

55. Buxton, W. and Myers, B. A study in two-handed input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1986), 321-326.

56. Card, S. K., Hong, L., Mackinlay, J. D. and Chi, E. H. 3Book: a 3D electronic smart book. In *Proceedings of the working conference on Advanced visual interfaces*, ACM (2004), 303-307.

57. Cockburn, A. Revisiting 2D vs 3D implications on spatial memory. In *Proceedings of the fifth conference on Australasian user interface - Volume 28*, Australian Computer Society, Inc. (2004), 25-31.

58. Cockburn, A., Ahlström, D. and Gutwin, C. Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. *International Journal of Human-Computer Studies*, *70*, 3 (2012), 218-233.

59. Cockburn, A., Gutwin, C. and Alexander, J. Faster document navigation with space-filling thumbnails. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2006), 1-10.

60. Cockburn, A. and McKenzie, B. Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2002), 203-210.

61. Conroy, K., Levin, D. and Guimbretière, F. Proofrite: A paper-augmented word processor. *Technical Report HCIL-2004-22, CS-TR-4652*, Human-Computer Interaction Lab, University of Maryland, 2004.

62. Cutler, L. D., Fröhlich, B. and Hanrahan, P. Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM (1997), 107-114.

63. Davis, M. R. and Ellis, T. O. The RAND tablet: a man-machine graphical communication device. *Memorandum RM-4122-ARPA*, RAND Corporation, Santa Monica, California, 1964.

64. De Nardi, A. Grafiti - Gesture Recognition mAnagement Framework for Interactive Tabletop Interfaces. *Master's Thesis*, University of Pisa, 2008.

65. Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, ACM (2001), 219-226.

66. Dimond, T. L. Devices for reading handwritten characters. In *Eastern Joint Computer Conference*, ACM (1958), 232-237.

67. Do-Lenh, S., Kaplan, F., Sharma, A. and Dillenbourg, P. Multi-finger interactions with papers on augmented tabletops. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, ACM (2009), 267-274.

68. Dourish, P. *Where the action is: the foundations of embodied interaction*. MIT Press, 2001.

69. http://tisch.sourceforge.net/

70. Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, *47*, 6 (1954), 381.

71. Fitzmaurice, G. W., Ishii, H. and Buxton, W. A. S. Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press/Addison-Wesley Publishing Co. (1995), 442-449.

72. Forlines, C., Wigdor, D., Shen, C. and Balakrishnan, R. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2007), 647-656.

73. Francik, E., Rudman, S. E., Cooper, D. and Levine, S. Putting innovation to work: adoption strategies for multimedia communication systems. *Communications of the ACM*, *34*, 12 (1991), 52-63.

74. Freeman, D., Benko, H., Morris, M. R. and Wigdor, D. ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2009), 165-172.

75. Frisch, M., Heydekorn, J. and Dachselt, R. Diagram editing on interactive displays using multi-touch and pen gestures. In *Proceedings of the 6th international conference on Diagrammatic representation and inference*, Springer-Verlag (2010), 182-196.

76. Frisch, M., Kleinau, S., Langner, R. and Dachselt, R. Grids & guides: multi-touch layout and alignment tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 1615-1618.

77. Frisch, M., Langner, R. and Dachselt, R. Neat: a set of flexible tools and gestures for layout tasks on interactive displays. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2011), 1-10.

78. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

79. Glushko, R. J. and McGrath, T. *Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services*. MIT Press, 2008.

80. Gray, E. Telautograph. *US Patent N° 386,815*, 1888.

81. Greanias, E. C., Guarnieri, C. R., Seeland Jr, J. J., Verrier, G. F. and Donaldson, R. L. Combined finger touch and stylus detection system for use on the viewing surface of a visual display device. *US Patent N° 4,686,332*, 1987.

82. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: the kinematic chain as a model. *Journal of Motor Behavior*, *19*, 4 (1987), 486-517.

83. Guimbretière, F. Paper augmented digital documents. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, ACM (2003), 51-60.

84. Haller, M., Brandl, P., Leithinger, D., Leitner, J., Seifried, T. and Billinghurst, M. Shared design space: sketching ideas using digital pens and a large augmented tabletop setup. In *Proceedings of the 16th international conference on Advances in Artificial Reality and Tele-Existence*, Springer-Verlag (2006), 185-196.

85. Hamilton, W., Kerne, A. and Robbins, T. High-performance pen + touch modality interactions: a real-time strategy game eSports context. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 309-318.

86. Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM (2005), 115-118.

87. Harryson, S. J. Anoto: Entrepreneurship from Creation to Commercialization. In *Know-Who Based Entrepreneurship: From Knowledge Creation to Business Implementation* (2007), 31-64.

88. Hinckley, K., Bi, X., Pahud, M. and Buxton, B. Informal information gathering techniques for active reading. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 1893-1896.

89. Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M. and Cutrell, E. The springboard: multiple modes in one spring-loaded control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2006), 181-190.

90. Hinckley, K., Pausch, R., Proffitt, D. and Kassell, N. F. Two-handed virtual manipulation. *ACM Transactions on Computer-Human Interaction*, *5*, 3 (1998), 260-302.

91. Hinckley, K., Pausch, R., Proffitt, D., Patten, J. and Kassell, N. Cooperative bimanual action. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, ACM (1997), 27-34.

92. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H. and Buxton, B. Manual deskterity: an exploration of simultaneous pen + touch direct input. In *Extended Abstracts of the 28th ACM Conference on Human factors in Computing Systems*, ACM (2010), 2793-2802.

93. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H. and Buxton, B. Pen + touch = new tools. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 27-36.

94. Hinckley, K., Zhao, S., Sarin, R., Baudisch, P., Cutrell, E., Shilman, M. and Tan, D. InkSeine: In Situ search for active note taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2007), 251-260.

95. Hinrichs, U., Hancock, M. S., Collins, C. and Carpendale, S. Examination of Text-Entry Methods for Tabletop Displays. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Tabletop'07)* (2007), 105-112.

96. Hoggan, E., Brewster, S. A. and Johnston, J. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 1573-1582.

97. Holman, D., Vertegaal, R., Altosaar, M., Troje, N. and Johns, D. Paper windows: interaction techniques for digital paper. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2005), 591-599.

98. Ishii, H. and Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1997), 234-241.

99. Jacob, R. J. K., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T. and Zigelbaum, J. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 201-210.

100. Javed, W., Kim, K., Ghani, S. and Elmqvist, N. Evaluating physical/virtual occlusion management techniques for horizontal displays. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction*, Springer-Verlag (2011), 391-408.

101. Jordà, S., Geiger, G., Alonso, M. and Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM (2007), 139-146.

102. Jorge, J. A. and Fonseca, M. J. A Simple Approach to Recognise Geometric Shapes Interactively. In *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances*, Springer-Verlag (2000), 266-276.

103. Kabbash, P., Buxton, W. and Sellen, A. Two-handed input in a compound task. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1994), 417-423.

104. Kabbash, P., MacKenzie, I. S. and Buxton, W. Human performance using computer input devices in the preferred and non-preferred hands. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, ACM (1993), 474-481.

105. http://www.tuio.org/?tuio20

106. Kaltenbrunner, M., Bovermann, T., Bencina, R. and Costanza, E. TUIO: A protocol for table-top tangible user interfaces. In *6th International Workshop on Gesture in Human-Computer Interaction and Simulation* (2005).

107. Kammer, D., Keck, M., Freitag, G. and Wacker, M. Taxonomy and overview of multi-touch frameworks: Architecture, scope and features. In *Proceedings of the Workshop on Engineering Patterns for Multitouch Interfaces* (2010).

108. Kammer, D., Wojdziak, J., Keck, M., Groh, R. and Taranko, S. Towards a formalization of multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2010), 49-58.

109. Kane, S. K., Avrahami, D., Wobbrock, J. O., Harrison, B., Rea, A. D., Philipose, M. and LaMarca, A. Bonfire: a nomadic system for hybrid laptop-tabletop interaction. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 129-138.

110. Kauranen, P. The ANOTO pen – Why light scattering matters. In *International Workshop on Microfactories* (2004).

111. Kay, A. C. A Personal Computer for Children of All Ages. In *Proceedings of the ACM annual conference*, ACM (1972).

112. Kessler, T. Can iWork on the iPad really compare to a Mac? *CNET Reviews MacFixIt*, 2012, http://reviews.cnet.com/8301-13727_7-57503839-263/can-iwork-on-the-ipad-really-compare-to-a-mac/.

113. Kin, K., Agrawala, M. and DeRose, T. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, Canadian Information Processing Society (2009), 119-124.

114. Koike, H., Sato, Y. and Kobayashi, Y. Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system. *ACM Transactions on Computer-Human Interaction*, *8*, 4 (2001), 307-322.

115. Kunz, A. and Fjeld, M. From Table–System to Tabletop: Integrating Technology into Interactive Surfaces. *Müller-Tomfelde (Ed.) Springer LNCS 6033* (2010), 53-72.

116. Kurtenbach, G., Fitzmaurice, G., Baudel, T. and Buxton, B. The design of a GUI paradigm based on tablets, two-hands, and transparency. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, ACM (1997), 35-42.

117. Lane, D. M., Napier, H. A., Peres, S. C. and Sándor, A. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, *18*, 2 (2005), 133-144.

118. Laufs, U., Ruff, C. and Zibuschka, J. Mt4j-a cross-platform multi-touch development framework. In *Proceedings of the Workshop of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (2010), 52-57.

119. Leganchuk, A., Zhai, S. and Buxton, W. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Transactions on Computer-Human Interaction*, *5*, 4 (1998), 326-359.

120. Leithinger, D. and Haller, M. Improving Menu Interaction for Cluttered Tabletop Setups with User-Drawn Path Menus. In *Proceedings of the Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems* (2007), 121-128.

121. Leitner, J., Powell, J., Brandl, P., Seifried, T., Haller, M., Dorray, B. and To, P. Flux: a tilting multi-touch and pen based surface. In *Extended Abstracts of the 27th ACM Conference on Human Factors in Computing Systems*, ACM (2009), 3211-3216.

122. Levy, D. M. *Scrolling Forward: Making Sense of Documents in the Digital Age*. Arcade Publishing, 2001.

123. Li, Y., Hinckley, K., Guan, Z. and Landay, J. A. Experimental analysis of mode switching techniques in pen-based user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2005), 461-470.

124. Li, Y., Landay, J. A., Guan, Z., Ren, X. and Dai, G. Sketching informal presentations. In *Proceedings of the 5th international conference on Multimodal interfaces*, ACM (2003), 234-241.

125. Liao, C., Guimbretière, F., Anderson, R., Linnell, N., Prince, C. and Razmov, V. PaperCP: exploring the integration of physical and digital affordances for active learning. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction*, Springer-Verlag (2007), 15-28.

126. Liao, C., Guimbretière, F. and Hinckley, K. PapierCraft: a command system for interactive paper. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM (2005), 241-244.

127. Liao, C., Guimbretière, F., Hinckley, K. and Hollan, J. Papiercraft: A gesture-based command system for interactive paper. *ACM Transactions on Computer-Human Interaction*, *14*, 4 (2008), 1-27.

128. Liao, C., Guimbretière, F. and Loeckenhoff, C. E. Pen-top feedback for paper-based interfaces. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM (2006), 201-210.

129. Liao, C., Tang, H., Liu, Q., Chiu, P. and Chen, F. FACT: fine-grained cross-media interaction with documents via a portable hybrid paper-laptop interface. In *Proceedings of the international conference on Multimedia*, ACM (2010), 361-370.

130. Liwicki, M., Rostanin, O., El-Neklawy, S. M. and Dengel, A. Touch & Write: a multi-touch table with pen-input. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, ACM (2010), 479-484.

131. Lopes, P., Mendes, D., Araújo, B. and Jorge, J. A. Combining bimanual manipulation and pen-based input for 3D modelling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, ACM (2011), 15-22.

132. MacKenzie, I. S., Sellen, A. and Buxton, W. A. S. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1991), 161-166.

133. Mackenzie, I. S. and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, *17* (2002), 147-198.

134. MacKenzie, I. S. and Tanaka-Ishii, K. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann Publishers Inc., 2007.

135. Marquardt, N., Jota, R., Greenberg, S. and Jorge, J. A. The Continuous Interaction Space: Interaction Techniques Unifying Touch and Gesture on and above a Digital Surface. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction*, Springer Berlin Heidelberg (2011), 461-476.

136. Masliah, M. R. and Milgram, P. Measuring the allocation of control in a 6 degree-of-freedom docking experiment. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM (2000), 25-32.

137. Matsushita, N., Ayatsuka, Y. and Rekimoto, J. Dual touch: a two-handed interface for pen-based PDAs. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM (2000), 211-212.

138. Matulic, F. Method and apparatus for creating document data, and computer program product. *US Patent N° 8,165,404.24*, 2012.

139. Matulic, F. SmartPublisher: document creation on pen-based systems via document element reuse. In *Proceedings of the 2006 ACM symposium on Document engineering*, ACM (2006), 182-183.

140. Matulic, F. and Norrie, M. Empirical evaluation of uni- and bimodal pen and touch interaction properties on digital tabletops. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ACM (2012), 143-152.

141. Matulic, F. and Norrie, M. Pen and Touch Gestural Environment for Document Editing on Interactive Tabletops. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, ACM (2013).

142. Matulic, F. and Norrie, M. C. Supporting active reading on pen and touch-operated tabletops. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ACM (2012), 612-619.

143. Matulic, F., Norrie, M. C., Kabary, I. A. and Schuldt, H. Gesture-supported document creation on pen and touch tabletops. In *Extended Abstracts of the 31st ACM Conference on Human Factors in Computing Systems*, ACM (2013), 1191-1196.

144. Moeller, J. and Kerne, A. ZeroTouch: an optical multi-touch and free-air interaction architecture. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 2165-2174.

145. Morris, M. R., Brush, A. J. B. and Meyers, B. Reading Revisited: Evaluating the Usability of Digital Display Surfaces for Active Reading Tasks. In *Proceedings of the Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2007*. (2007).

146. Norman, D. A. The Way I See It: Signifiers, not affordances, *Interactions - Designing games: why and how*, 2008.

147. Norrie, M. C., Palinginis, A. and Signer, B. Content publishing framework for interactive paper documents. In *Proceedings of the 2005 ACM symposium on Document engineering*, ACM (2005), 187-196.

148. O'Hara, K. and Sellen, A. A comparison of reading paper and on-line documents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, ACM (1997), 335-342.

149. Otlet, P. *Traité de documentation: le livre sur le livre, theéorie et pratique*. Editiones Mundaneum, 1934.

150. Papadopoulou, S., Ignat, C., Oster, G. and Norrie, M. Increasing awareness in collaborative authoring through edit profiling. In *Proceedings of the IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing*, IEEE (2006), 1-9.

151. Parker, I. Absolute PowerPoint: Can a Software Package Edit our Thoughts?, *The New Yorker*, 2001.

152. Pastel, R. Measuring the difficulty of steering through corners. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2006), 1087-1096.

153. Qin, Y., Yu, C., Jiang, H., Wu, C. and Shi, Y. pPen: enabling authenticated pen and touch interaction on tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2010), 283-284.

154. Raisamo, R. An alternative way of drawing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM (1999), 175-182.

155. Raisamo, R. and Räihä, K.-J. A new direct manipulation technique for aligning objects in drawing programs. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, ACM (1996), 157-164.

156. Ramakers, R., Vanacken, D., Luyten, K., Coninx, K. and Schönning, J. Carpus: a non-intrusive user identification technique for interactive surfaces. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 35-44.

157. Ramanahally, P., Gilbert, S., Niedzielski, T., Velázquez, D. and Anagnost, C. Sparsh UI: A Multi-Touch Framework for Collaboration and Modular Gesture Recognition. In *Proceedings of the World Conference on Innovative Virtual Reality*, ASME (2009), 137-142.

158. Ramos, G., Robertson, G., Czerwinski, M., Tan, D., Baudisch, P., Hinckley, K. and Agrawala, M. Tumble! Splat! helping users access and manipulate occluded content in 2D drawings. In *Proceedings of the working conference on Advanced visual interfaces*, ACM (2006), 428-435.

159. Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L. and Fuchs, H. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM (1998), 179-188.

160. Raskin, J. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., 2000.

161. Rekimoto, J. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2002), 113-120.

162. Rekimoto, J. and Saitoh, M. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM (1999), 378-385.

163. Richter, S., Holz, C. and Baudisch, P. Bootstrapper: recognizing tabletop users by their shoes. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 1249-1252.

164. Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D. and Dantzich, M. v. Data mountain: using spatial memory for document management. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, ACM (1998), 153-162.

165. Robertson, G., Dantzich, M. v., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. and Gorokhovsky, V. The Task Gallery: a 3D window manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2000), 494-501.

166. Ryall, K., Forlines, C., Shen, C., Morris, M. R. and Everitt, K. Experiences with and Observations of Direct-Touch Tabletops. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, IEEE Computer Society (2006), 89-96.

167. Sandee Adobe Creative Team Working with Comments. In *Adobe Acrobat X Classroom in a Book*, Adobe Press (2010), 221-224.

168. Schilit, B. N., Golovchinsky, G. and Price, M. N. Beyond paper: supporting active reading with free form digital ink annotations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press/Addison-Wesley Publishing Co. (1998), 249-256.

169. Schmidt, D., Chong, M. K. and Gellersen, H. HandsDown: hand-contour-based user identification for interactive surfaces. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction*, ACM (2010), 432-441.

170. Scholliers, C., Hoste, L., Signer, B. and Meuter, W. D. Midas: a declarative multi-touch interaction framework. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, ACM (2011), 49-56.

171. Sculley, J. and Byrne, J. A. *Odyssey: Pepsi to Apple : A Journey of Adventure, Ideas, and the Future*. Harpercollins, 1987.

172. Sellen, A. J. and Harper, R. H. R. *The Myth of the Paperless Office*. MIT Press, 2001.

173. Sellen, A. J., Kurtenbach, G. P. and Buxton, W. A. S. The prevention of mode errors through sensory feedback. *Human-Computer Interaction*, *7*, 2 (1992), 141-164.

174. Shaw, C. and Green, M. Two-handed polygonal surface design. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, ACM (1994), 205-212.

175. Signer, B. Fundamental Concepts for Interactive Paper and Cross-Media. *PhD Thesis*, ETH Zurich, 2006.

176. Signer, B. and Norrie, M. C. PaperPoint: a paper-based presentation and interactive paper prototyping tool. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM (2007), 57-64.

177. Song, H., Grossman, T., Fitzmaurice, G., Guimbretière, F., Khan, A., Attar, R. and Kurtenbach, G. PenLight: combining a mobile projector and a digital pen for dynamic

visual overlay. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 143-152.

178. Song, H., Guimbretière, F., Grossman, T. and Fitzmaurice, G. MouseLight: bimanual interactions on digital paper using a pen and a spatially-aware mobile projector. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 2451-2460.

179. Steimle, J. Survey of Pen-and-Paper Computing. In *Pen-and-Paper User Interfaces*, Springer (2012), 19-65.

180. Streitz, N. A., Geißler, J. and Holmer, T. Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces. In *Proceedings of the First International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, Springer-Verlag (1998), 4-21.

181. Streitz, N. A., Tandler, P., Müller-Tomfelde, C. and Konomi, S. Roomware: Towards the Next Generation of Human-Computer Interaction based on an Integrated Design of Real and Virtual Worlds. In *Human-Computer Interaction in the New Millennium*, Addison-Wesley (2001), 553-578.

182. Subramanian, S., Aliakseyeu, D. and Lucero, A. Multi-layer interaction for digital tables. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM (2006), 269-272.

183. Summet, J., Abowd, G. D., Corso, G. M. and Rehg, J. M. Virtual rear projection: do shadows matter? In *Extended Abstracts of the 23rd ACM Conference on Human Factors in Computing Systems*, ACM (2005), 1997-2000.

184. Sutherland, I. E. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, ACM (1964), 6.329-326.346.

185. Suzuki, Y., Misue, K. and Tanaka, J. Stylus enhancement to enrich interaction with computers. In *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, Springer-Verlag (2007), 133-142.

186. Swinnen, S. P. and Wenderoth, N. Two hands, one brain: cognitive neuroscience of bimanual skill. *Trends in cognitive sciences*, *8*, 1 (2004), 18-25.

187. Tashman, C. S. and Edwards, W. K. Active reading and its discontents: the situations, problems and ideas of readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 2927-2936.

188. Tashman, C. S. and Edwards, W. K. LiquidText: a flexible, multitouch environment to support active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 3285-3294.

189. Tian, F., Xu, L., Wang, H., Zhang, X., Liu, Y., Setlur, V. and Dai, G. Tilt menu: using the 3D orientation information of pen devices to extend the selection capability of pen-based user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 1371-1380.

190. Tognazzini, B. The "Starfire" video prototype project: a case history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1994), 99-105.

191. Tu, H., Ren, X. and Zhai, S. A comparative evaluation of finger and pen stroke gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2012), 1287-1296.

192. Ullmer, B. and Ishii, H. The metaDESK: models and prototypes for tangible user interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, ACM (1997), 223-232.

193. Walny, J., Lee, B., Johns, P., Riche, N. H. and Carpendale, S. Understanding Pen and Touch Interaction for Data Exploration on Interactive Whiteboards. *IEEE Transactions on Visualization and Computer Graphics*, *18*, 12 (2012), 2779-2788.

194. Wang, F., Cao, X., Ren, X. and Irani, P. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 23-32.

195. Wang, F. and Ren, X. Empirical evaluation for finger input properties in multi-touch interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2009), 1063-1072.

196. Weibel, N., Ispas, A., Signer, B. and Norrie, M. C. Paperproof: a paper-digital proof-editing system. In *Extended Abstracts of the 26th Conference on Human Factors in Computing Systems*, ACM (2008), 2349-2354.

197. Weiss, M., Voelker, S., Sutter, C. and Borchers, J. BendDesk: dragging across the curve. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2010), 1-10.

198. Wellner, P. Interacting with paper on the DigitalDesk. *Communications of the ACM*, *36*, 7 (1993), 87-96.

199. Westerman, W. Hand tracking, finger identification, and chordic manipulation on a multi-touch surface. *PhD Thesis*, University of Delaware, 1999.

200. Wigdor, D., Benko, H., Pella, J., Lombardo, J. and Williams, S. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 1581-1590.

201. Wilcox, L. D., Schilit, B. N. and Sawhney, N. Dynomite: a dynamically organized ink and audio notebook. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, ACM (1997), 186-193.

202. Wilson, A. D. PlayAnywhere: a compact interactive tabletop projection-vision system. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM (2005), 83-92.

203. Wilson, A. D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 273-282.

204. Wimmer, R., Hennecke, F., Schulz, F., Boring, S., Butz, A. and Hußmann, H. Curve: revisiting the digital desk. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, ACM (2010), 561-570.

205. Wobbrock, J. O., Wilson, A. D. and Li, Y. 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*, ACM (2007), 159-168.

206. Wu, C.-S., Robinson, S. J. and Mazalek, A. Turning a page on the digital annotation of physical books. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, ACM (2008), 109-116.

207. Wu, M. and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, ACM (2003), 193-202.

208. Wu, M., Chia, S., Ryall, K., Forlines, C. and Balakrishnan, R. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems* (2006), 185-192.

209. Yee, K.-P. Two-handed interaction on a tablet display. In *Extended Abstracts of the 22nd Conference on Human Factors in Computing Systems*, ACM (2004), 1493-1496.

210. Yeh, R., Liao, C., Klemmer, S., Guimbretière, F., Lee, B., Kakaradov, B., Stamberger, J. and Paepcke, A. ButterflyNet: a mobile capture and access system for field biology research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2006), 571-580.

211. Yeh, R. B., Paepcke, A. and Klemmer, S. R. Iterative design and evaluation of an event architecture for pen-and-paper interfaces. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM (2008), 111-120.

212. Zabramski, S. Careless touch: a comparative evaluation of mouse, pen, and touch input in shape tracing task. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, ACM (2011), 329-332.

213. Zeleznik, R., Bragdon, A., Adeputra, F. and Ko, H.-S. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 17-26.

214. Zeleznik, R. C., Forsberg, A. S. and Strauss, P. S. Two pointer input for 3D interaction. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM (1997), 115-120.

215. Zucco, J. E., Thomas, B. H. and Grimmer, K. Evaluation of Three Wearable Computer Pointing Devices for Selection Tasks. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, IEEE Computer Society (2005), 178-185.