HELSINKI UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

Information Networks Degree Program

Roope Rainisto

TEXT INPUT ON MOBILE DEVICES:

DESIGNING A TOUCH SCREEN INPUT METHOD

Master's Thesis

Helsinki, May 22, 2007

Supervisor: Professor Marko Nieminen, D.Sc. (Tech.)

Instructor: Jussi-Pekka Kekki, MA

HELSINKI UNIVERSITY OF TECHNOLOGY ABSTRACT OF MASTER'S THESIS

Department of Computer Science and Engineering
Information Networks Degree Program

Author:	Roope Rainisto		
Title:	Text input on mobile devices: designing a touch screen input method		
Pages:	84	**Date:**	22.05.2007
Professorship:	Human Centered Information Systems	**Professorship code:**	T-121
Supervisor:	Professor Marko Nieminen, D.Sc. (Tech.)		
Instructor:	Jussi-Pekka Kekki, MA		

Mobile communication devices, especially mobile phones, live in a field of extensive research and development. The latest generation of mobile devices can provide a rich set of communication and information management features for their users.

One critical aspect of mobile device design relates to the design of its text input methods: how to provide the best possible text input experience on a device with both physical and performance limitations.

The theoretical part of this thesis discusses the main issues relating to text input on mobile devices. Popular input methods and their designs are introduced and discussed. Special focus is put to the questions relating to touch screen versus hardware key usage, and to the use of predictive and assistive user interface methods in text input.

The practical part of the thesis describes the work performed when designing the user interaction and interface for one such text input method: a virtual keyboard design for a new touch screen based wireless mobile device. Part of the work is based on improving the text input method design found in an earlier product from the same company. The merits and the problem areas of the previous design are analyzed.

The thesis concludes with a discussion on the main topics and findings of the study. Some thoughts on general input method design principles and further research areas are given. A summary of known usability test results for the produced design is also presented.

Keywords: Text input, virtual keyboard, touch screen, mobile device

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tietotekniikan osasto
Informaatioverkostojen koulutusohjelma

Tekijä:	Roope Rainisto		
Työn nimi:	Tekstinsyöttö mobiililaitteella: kosketusnäytölle soveltuvan tekstinsyöttömetodin suunnittelu		
Sivumäärä:	84	**Päiväys:**	22.05.2007
Professuuri:	Ihmisläheiset tietojärjestelmät	**Koodi:**	T-121
Työn valvoja:	Professori Marko Nieminen, TkT		
Työn ohjaaja:	Jussi-Pekka Kekki, FM		

Mobiilit kommunikaatiolaitteet, etenkin matkapuhelimet, ovat tällä hetkellä voimakkaan tutkimuksen ja kehityksen kohteena. Mobiililaitteiden uusin sukupolvi kykenee tarjoamaan monipuolisen kirjon erilaisia kommunikaatioon ja informaation hallintaan liittyviä ominaisuuksia käyttäjilleen.

Yksi mobiililaitteiden suunnittelun tärkeä osa-alue liittyy sen tekstinsyöttömetodien suunnitteluun: kuinka tarjota paras mahdollinen tekstinsyötön käyttökokemus laitteella, jolla on lukuisia rajoitteita sekä fyysisten ominaisuuksiensa että suorituskyvynsä suhteen.

Diplomityön teoreettinen osuus kuvaa mobiililaitteen tekstinsyöttöön liittyviä tärkeimpiä teoreettisia taustoja, tutkimuksia ja nykyisiä menetelmiä. Erityishuomiota tutkimuksessa asetetaan kosketusnäytön ja fyysisten näppäinten käytön välisiin eroihin, ja ennustavien ja käyttäjää avustavien käyttöliittymätoimintojen tarkasteluun.

Diplomityön käytännön osuus kuvaa erään mobiililaitteen kosketusnäytölle tarkoitetun virtuaalinäppäimistöön pohjaavan tekstinsyöttömetodin käyttöliittymän suunnitteluprosessin. Osa työstä perustuu kohdetta edeltävän tuotteen vastaavan metodin suunnitelmiin, sen hyvien piirteiden hyödyntämiseen ja puutteiden parantamiseen.

Diplomityö päättyy pohdintaosuuteen, jossa esitetään ajatuksia kerätystä tietämyksestä ja annetaan ajatuksia jatkokehityksen ja –tutkimuksen kohteiksi. Työssä esitetään myös lyhyt yhteenveto suunniteltuun tekstinsyöttömetodiin liittyvistä käytetävyystestituloksista.

Avainsanat: Tekstinsyöttö, virtuaalinäppämistö, kosketusnäyttö, mobiililaite

Preface

I consider myself very fortunate to have the opportunity to be involved in this project. Projects like these do not happen every day. Stepping outside the secure playground of the academic world and into the murky waters of real life commercial development has definitely opened my eyes to a different reality, one more hectic, interesting and colourful than what any schoolbook can describe.

Writing this thesis has been possible only due to the kind help, support and contributions of a large number of people, all of whom deserve my gratitude. I would like specifically to thank my professor Marko Nieminen for his positive attitude, experience, and personal guidance in creating this thesis, and my thesis instructor Jussi-Pekka Kekki for his detailed feedback and support throughout the project. Additional thanks go out also to Tomi Rauste from Movial Corporation. Creating this thesis would not have been possible without the combined efforts of these individuals.

I would also like to thank Kalle Saarinen for supporting and giving design guidance for the UI design project under its many various stages, as well as all of my other managers in Nokia for their trust and support. Chen Xun and the entire software team assisted in their effort and knowledge in helping to keep the proposed designs realistic and relevant to the actual implementation challenges. Dr. Simo Säde provided his insight and guidance many times during the design process. Special thanks go out to Miika Silfverberg for his review and valuable feedback provided for this master's thesis.

I would also like to thank all of my fellow interaction design team members for providing their invaluable support and feedback, as well as making coming to work each day a fun and enjoyable experience.

And finally, to all the people in my personal life, and my friends and family, to all those I care about: the days we laugh and the days we cry, the good times and the bad times, to me they're all just as good.

In Helsinki, May 2007 Roope Rainisto

Table of contents

Acronyms and terms

Cursive input	A form of handwritten text where words are written in a manner where each letter is not formed separately but rather the whole word is drawn as a continuous stroke.
Graffiti	Text input mechanism used in many Palm PDA devices. Each character has a unique handwriting pattern, designed for quick input. These patterns do not necessarily match real-world character shapes.
GTK+	A multi-platform toolkit for creating graphical user interfaces. GTK+ is free software and part of the GNU Project.
HWR	Handwriting recognition. A method of text input where the user enters text by drawing patterns, usually resembling the shapes of natural characters. The recognition system converts these patterns into characters and words.
ITU-T	The ITU-T E.161 keypad. This is a phone keypad layout from the International Telecommunication Union. ITU-T is used to describe a commonly utilized 12-key phone keypad layout, containing numbers 1-0 and * and #.
KSPC	Keystrokes per character – how many keystrokes on average is required to produce one character in an input method.
Multi-tap	Text input mechanism, common in many mobile phones, where multiple characters are associated with one hardware key. Pressing the key once outputs the first character, pressing it twice outputs the second character etc.
Qwerty	A keyboard layout commonly found in desktop computers used in western languages. Named after the first six characters found on the top character row of the keyboard.
PDA	Personal Digital Assistant. A mobile device used for managing personal information. The device usually includes one or more PIM applications.
PIM	Personal Information Management. Common PIM applications include the calendar, address book or note taking applications, in a larger sense all applications that are used for managing the personal information of a user.

T9 A predictive text input method for mobile devices from Tegic Communications.

VKB Virtual keyboard. An on-screen representation of a physical keyboard. Text input is performed by pressing the virtual buttons displayed on the screen.

1. INTRODUCTION

1.1 Background

One of the biggest advances in the 20th century came about through the development of microprocessors. Technological progress in miniaturization brought computing first into the desktop and then soon to even smaller, physically portable form factors. In little over 20 years mobile portable technology for personal devices has advanced dramatically - from its humble beginnings as calculators and electronic organizers - to the current state of the art, where seemingly little fundamental restrictions on what it is capable of are left remaining.

Whereas the rate of technological advance has been dramatic on the hardware and the performance of mobile devices, the interaction layer between man and machine has not developed to the same extent. A mobile device can perform millions of complex calculations per second, but it is very much more limited in the manner how its users can give input and read output from this device. One fundamental type of input is the input of text and numbers into the operating system. The topic of mobile text input has been heavily studied and researched in the past decades, because of its importance and of the challenges and possibilities that lie ahead.

Although the text input methods have potential to develop also in desktop environments, the need to find better solutions is substantially greater with portable devices than with fixed terminals, where the current solutions already give reasonable performance for most users. Desktop environments offer large displays, the use of mouse and full sized keyboards and almost instantaneous responses after any user actions, and they therefore set the expectations of the end users extremely high. Even a mobile phone could also offer compelling user experiences with rich communication, information retrieval and personal information management, but at the same time their users often find the inputting of even short messages or queries frustrating and laborious, severely hampering the potential usefulness of these kinds of devices.

This is the technological climate that we all now live in, and this document serves as one artefact of this reality. Another part of the story behind this thesis starts in early 2004. Nokia had previously started a project to design and produce a new type of device: a mobile wireless internet tablet device (now known as the Nokia 770 Internet Tablet), the first device utilizing

a new operating system created by Nokia, created mainly from open source software components.

I joined the project as an interaction designer in January 2004. As one of the members in a team of interaction designers, one of the tasks that I was given was to start studying issues relating to text input on mobile devices. This task ultimately grew into specifying the user interaction and the user interface for the various virtual text input methods of the device. This design work also helped to form the practical part of this thesis. To be able to perform this task, the theoretical research part then came naturally: what kind of issues does one need to consider when designing such an input method.

During the design work, the interaction design team had access to the design documents of another previous Nokia product: the Nokia 7700 Series 90 touch widescreen smartphone. Therefore we had the opportunity to specify parts of our UI design based on already available documents and usability knowledge, and further iterate the design on the aspects where we felt that improvements should and could realistically be done. Due to a different technical architecture, we had both the need and the opportunity to re-design and re-implement critical parts of the operating system and the user interface that it presents to the user.

This thesis is written as post mortem document: it examines the work done retrospectively. The design work was done before this thesis was completed, as is often the case.

1.2 Objectives of the study

This thesis focuses on the needs of the users of being able to input text with a mobile device in an efficient manner. As a practical consideration, at the beginning of this thesis the development work on a new touch screen mobile device had been started. This device was lacking a design for its text input methods: this sparked the need to study the issue in a detailed manner.

The thesis work is split into a theoretical study and a practical design task.

The study part of this thesis is split into two objectives. Firstly, a general overview on the level of existing research in the field of mobile text input is performed. Factors and issues that

influence the interaction design of a mobile device with text input capabilities are introduced and discussed. This study is done as a construction, to build a model of knowledge about the different elements in the field of mobile text input.

The second study objective is to describe and evaluate the design of the text input methods in the baseline device – a Nokia touch screen smartphone – that was available at the start of the project: the strengths and weaknesses of its text input method design are analyzed and discussed.

The main practical objective in this thesis is a classic product development task of designing a usable and functional text input method within given limitations for a touch screen mobile device. The design utilizes the previously constructed research knowledge as well as improves on the problem issues found in the Nokia 7710 baseline design. The design process is described and evaluated: motivations behind the major changes done to the previous design are described and discussed. The completed design is briefly described and a summary of the known usability study results is given.

Finally, discussion items are raised for future research and design improvements.

1.3 Scope and limitations of the study

The research field for text input devices is very extensive and impossible to cover adequately in one master's thesis. Therefore the study will by scoped by making the following limitations:

Text input can be performed with many kinds of devices, both static and mobile. As the first limitation this study focuses mainly on mobile text input devices. Mobile devices by their nature contain many restrictions and design challenges, caused by the physical limitations created by the device and by the varying surroundings where the device will be utilized. Text input issues related to desktop (or laptop) computing are mainly scoped outside this study.

For a mobile device the current two main input mechanisms are the use of device hardware keys and the use of the device touch screen. This thesis focuses on the interaction challenges created by providing and utilizing a touch screen for text input, although some issues with hardware key text input are also discussed. In addition to these two, additional text input

mechanisms for a mobile device do certainly exist: for example physical interaction can be replaced by text input by speech recognition. These alternative mechanisms are also scoped outside the main focus areas in this thesis.

Inside the scope of touch screen input mechanisms, many different kinds of designs for text input user interfaces also exist. The main focus will be on the design of having a virtual keyboard for text input, and on the issues that come from utilizing it. A wide range of alternative methods, such as handwriting recognition, exist, but they are not discussed in this thesis.

Another limitation of this study is to focus on the use of western languages, and those written using the Latin alphabet. Alphanumeric input in western languages shares for the most parts the similar design challenges across the language set. Chinese, Japanese and Arabic languages are examples of languages that are scoped outside this study, since they contain a whole new set of issues for touch screen text input.

Furthermore, many advances are done in the research field of text input for disabled users. This is yet another very interesting field of research, but is also scoped outside for the purposes of this study, since it would change the research focus of this thesis substantially.

1.4 Structure of the thesis

The structure of the master's thesis is as follows:

After the introductory chapter, chapter 2 starts with a research and literature study of the problem areas relating to mobile text input. Here research and theoretical issues most relevant to the task of designing a touch screen virtual keyboard text input method are discussed.

Chapter 3 continues the background work by describing the previous baseline design that was available at the start of the project. An analysis of the text input methods in this previous design is provided, and its strengths and weaknesses are discussed.

Chapter 4 moves into the practical side of the thesis by describing the actual user interface design work. It discusses the principles and the major changes done in the text input design,

referring to the information gathered from the literature study and the baseline design evaluation when available.

Finally, chapter 5 evaluates the finished design by giving a summary of the usability study results, as well as by providing conclusions and discussions about the project and the issues at hand. Some ideas and questions for further research in the field of text input are also raised.

2. PRINCIPLES OF MOBILE TEXT INPUT

This chapter presents a summary of the current research field on key issues related to mobile text input and virtual keyboard usage. Since the research field is extensive, this chapter focuses on the research subjects and key issues that hopefully prove most relevant to the practical text input method design project at hand.

2.1 Text input in mobile devices

The history of mobile devices with text input capabilities depends on the definition of such devices. One can for instance claim that nearly all commercially available typewriters are examples of mobile text input devices, since they are essentially portable and allow text input to be performed in a sensible manner.

If the discussion is scoped to more relevant terms, 20th century advances in transistor technology brought the first truly mass market mobile device with input capabilities: the electronic pocket calculator. Although pocket calculator input mainly consists of numbers instead of characters, and the user interface is usually more straightforward and dedicated to a single purpose, most of the same fundamental issues and problems are still present. After pocket calculators, both PDA devices and mobile phones began to gain popularity. One of the forerunners of commercial PDA mobile devices featuring touch screens and handwriting recognition was the Apple Newton line of devices. The Apple Newton devices featured no hardware keys for text input, instead the device touch screen provided the user text input through a virtual keyboard or a handwriting recognition feature. The 1990's brought a surge of PDA devices to the marketplace. Well known PDA device developers include Palm, Research In Motion and Hewlett-Packard.

If analyzing the market by device volumes, the mobile phone is currently clearly most popular type of mobile text input device. Mobile phones originally started by providing only voice calling capabilities, but they rather quickly expanded into messaging and other functions. The current feature set of advanced mobile phones (so-called 'multimedia computers' and 'smartphones') provide a large range of different functions and services for

their users. Smartphones and other wireless communication devices are making the market of pure non-connected PDA devices rather quickly obsolete.

Mobile devices, as their name suggests, differ from desktop devices essentially in their mobility. The device is used in a wide range of environments and conditions, some of which are difficult to anticipate when designing the device. Even basic properties such as light, temperature, noise levels etc. can vary considerably in different environments. Whereas a desktop device is usually used for a longer period of a time at once, mobile device use is usually temporal and occasional. Including communication features in mobile devices lead to the situation that the device needs to be utilized also when the user had had no intention of using it: for instance to answer to an unexpected incoming voice call. The user can be very much unprepared for utilizing the device in these kinds of cases, both mentally and physically.

Mobile devices generally are constrained in one or more technical or physical factors. In order to achieve portability and a long battery life, the CPU performance and mobile storage capacity are usually limited and the screen size is small. The same limitations also apply to the amount of keys and other physical controls available on the device. Network communication speeds, usually through wireless networking, rarely reach levels similar to desktop computing. Mobile devices run mostly on batteries and therefore require constant recharging.

However, technological advances are leading towards the situation where the key constraints of a mobile device are not related to its cost or technical performance but rather to the more fundamental constraints relating to the physical properties of the device and of the device user. The technical performance of a mobile device will always relatively lag behind static or desktop device performance - because of the additional costs and efforts related to miniaturization - but even the current performance allow mobile devices to perform a wide range of complex tasks.

Device mobility combined with text input brings up many interesting design challenges. To take one example, a typical and unique use case for a mobile device involves text input while physically moving, i.e. walking around. Research shows that input speeds are lower walking than when standing still, and simultaneous text input also has the noted tendency to slow the

walking speed of users down from their normal walking speeds. On the other hand, the speed of the walking does not seem to affect the text input speeds, only the fact that the person is walking in general, and there seems to be no correlation between the difficulty of the input task and the walking speed of the user. (Mizobuchi et al., 2005)

Some of these design challenges have no significant current counterparts in the desktop environment, making mobile text input an interesting and important field of design and study.

2.2 Methods of mobile text input

When considering text input methods on a mobile device, Poika Isokoski categorizes mobile text input method design into two basic approaches: selection and recognition methods. Selection methods display the available input capabilities to the user directly and explicitly, whereas with recognition methods the user is under the illusion that the system is able to recognize more freely formatted input. A virtual keyboard is an example of a selection approach, and handwriting recognition is an example of the recognition approach to input. More complex systems Isokoski categorizes as composite systems, which contain then parts of multiple approaches or techniques, in serial or parallel mechanisms. (Isokoski, 2004)

The majority of the current popular mobile text input methods can be categorized to be selection methods, and most of these are based on the use of hardware buttons on the device. Since mobile devices are most often too small to fully fit a keyboard layout such as Qwerty – i.e. to provide one key for each required character - various designs exist to bring the amount of required buttons down to a smaller number. For instance the Nokia Communicator 9300 phone opens up to reveal a “full keyboard” of 58 character keys, whereas a normal PC Qwerty layout keyboard usually approximately 100 various keys.

Most mobile phones currently base their text input methods on a 12 key physical phone keypad, (with numbers 1-0, *, #). This design is commonly referred to as the ITU-T keypad (see Figure 1). Mobile phones inherited this layout mostly from conventional landline phones. Letters have been added over the 2-9 keys, so that each button contains multiple characters. The alphabets have been used to make numbers easier to remember; for instance,

a company could have had a phone number like *1-800-227-738464*, which could have then been marketed with a friendly name "*1-800-CARPETING*".

Figure 1 - The ITU-T E.161 keypad

On mobile phones, one of the most common text input methods using ITU-T is the multi-tap input method. In multi-tap input text input is performed by pressing a character button the one or more times, depending on the position on which the character is located in the button. For instance button 2 contains the characters A, B and C; A is inserted by tapping the button once, B by tapping twice and C by tapping it three times. If two letters of the same key are to be entered, the first character is committed either by waiting for a short duration (timeout) or by committing the first character on screen by pressing another control key on the device.

The layout of the keys in the ITU-T model is based on alphabetical ordering. A theoretically more efficient layout (minimizing the required amount of key presses) would reorder the characters so that they would follow the frequencies of character use in each particular language, so that the most frequently used characters would be available with only one button press. Then again, this optimized ordering would be need different in each language, since the frequency of how various characters are utilized differs between various languages.

Alternatives to the ITU-T layout naturally exist. For example some of the Blackberry devices from Research in Motion feature a 14 button keyboard layout (called the SureType method), where each button contains two characters instead of three characters, as found in ITU-T. (Segan, 2007)

A major alternative to text input via hardware keys is input through the touch screen of a device. Touch screens are gaining in popularity, as their technology becomes cheaper to

manufacture and as mobile devices and their operating systems keep gaining new functionalities. Touch screen allows the developer to design a fully flexible user interface: whereas with hard keys the same hard keys need to be used in all functions of the device, with a touch screen different controls can be provided to the user for each different task or function. Two popular touch screen input methods are the virtual keyboard and the handwriting recognition methods.

Virtual keyboard is an input method that displays a representation of a physical keyboard or keypad on the device touch screen. Basic usage is still fundamentally similar to a physical keyboard: text input is performed by tapping the buttons displayed on the keyboard. Handwriting recognition is an input method where the user inputs text by writing patterns, most often resembling the local natural alphabet, on the touch screen, and then the handwriting recognition engine converts those patterns into words and characters. Basic handwriting systems recognize words by looking for individual characters, more advanced systems can also recognize cursive input (and whole words). The handwriting styles of individual persons vary to a great extent, both in relation to other persons and even within one person: the variation in how a person writes some particular word in different times and circumstances can be significant. This causes extensive challenges for the recognition engine to be able to accurately recognize handwritten text. Advanced handwriting recognition methods are generally adaptive, learning from the users and allowing them to train their custom handwriting styles to the device, potentially reducing the amount of errors and increasing the overall recognition rate.

Next to the hard key and touch screen input methods, alternative input methods naturally do exist, but they have not yet gained widespread popularity. Speech recognition would be a natural and potentially very fast form of text input, but it faces many problems, both technical and conceptual. Technically mobile devices can not yet offer adequate performance in either the speed or the recognition quality to make speech recognition a competitive input method; even desktop devices are barely to provide accurate recognition, and only in optimal settings and environments.

Several problems also exist with speech recognition on a conceptual level. A rather fundamental issue is the general aversion that humans have in talking to machines. Input by speech is very error prone, mobile environments can often be noisy and full of distractions,

and therefore not fully suitable for speech recognition. Also achieving privacy with speech recognition is not properly possible in environments with other people. Speech recognition is often tested by having the test subjects read aloud a predefined block of text. In real life, text input is often a task where the user is not completely sure what he or she is intending to say. Sentences are formed iteratively, with extensive editing, hesitation and reformulating, and this does not suite speech recognition at all. Especially in a mobile device the editing and error correction of the recognized speech becomes very difficult: commanding the editing through speech is very cumbersome, or if done by other means, starts to negate the whole value of having a speech recognition method available.

Another strategy for text input is the abbreviated input. A common example of such a method is stenography, which is a shorthand writing style, still used in certain occasions, for instance in medical or legal courtroom situations. As one research example, Shieber and Baker (2003) describe a scheme where user inputs the consonants of each word, adding the vowels only when absolutely necessary. Their argument is that with the normal use of predictive (word completion) methods the user needs to constantly observe and select the predictive candidates and that this significantly increases the cognitive load for the user. With the use of abbreviated input similar monitoring isn't required from the user, as long as the system is able to successfully decode the abbreviated input. In their tests the model where vowels are omitted results in about 26.5% less characters needing to be inputted, with a 3.0% decoding error rate from the system. They also demonstrate 1.5% error rate would be obtainable with the use of an engine that would adapt and learn from later occurrences of previously unknown words. (Shieber & Baker, 2003)

It is noteworthy to mention that whereas the use of abbreviated input is a relatively unfamiliar concept with the western Latin languages, many Asian text input methods, such as PinYin for Chinese, can provide and take advantage of these principles. For instance some PinYin input method implementations offer the possibility of "initials-only input", where the user inputs only the first character of the sound of each Chinese character and the system then provides candidates for the full words and sentences.

Chapter 2.9 of this thesis discusses the use of predictive input methods in western languages. The T9 input method is one well known example of such a predictive input method.

2.3 Learning processes for text input methods

An important factor in mobile text input methods relates to the learnabilty of such methods. Virtually any human designed system can be learned, with sufficient time and practice. With human lifetimes being finite, and the time that a person gets to spend with a mobile device even more so, it is extremely useful if the input method design does not require a lengthy learning period, and also that any new learned interaction would be retained for a long period.

A simple utilitarian model would suggest that if the total efficiency gained and time saved (during the lifetime of usage) of using an unfamiliar input method would be greater than the time and effort that learning such an input method takes, then that it would be useful to learn that method.

On the other hand, humans are usually not able to perform such complex calculations, nor they can know the total lifetime of usage of a particular input method. If an input method seems too complex to learn, most users will simply not use it. If such complex input methods are the only input methods available on a device, this might lead to users not acquiring or using the entire device at all. Additionally, learning a new input method can have a negative effect (retroactive influence) on previously learned methods and patterns: for instance learning the Dvorak keyboard layout will negatively influence the Qwerty layout skills a user might have.

Scott MacKenzie et al. write that the learning process of a text input system has three distinct phases: Discovery phase, motor reflex acquisition phase and the terminal (Fitts' law) phase. In the discovery phase users start to discover the mechanics of the input method. Input speeds at this stage are dominated by the users' familiarity with conventions, such as alphabetic ordering or other previously learned models. From the discovery phase the users quickly move into the motor reflex acquisition phase, where the speed of input starts to increase logarithmically. This phase lasts for thousands of keystrokes. The final phase is the terminal phase, where the users have grown experts with the input method and the input speeds start to reach close to their theoretical maximums. Reasonable estimates of these maximum speeds can be calculated with various theoretical models, where motor constraints,

physical distances, reaction times, amount of keystrokes required etc. are calculated. (MacKenzie et al, 2001a) This learning process assumes that the text input system question is essentially fixed, that is the users can learn the positions and the behaviour of the input controls in a predictable manner.

A related factor is the time it takes for a user to learn an input method. For instance when talking about a virtual keyboard input method, there is a very high likelihood that when the user comes across a virtual keyboard, he can already utilize knowledge of previous physical and other virtual keyboards that he might have previously used. Initially learning to utilize a keyboard, to learn its mechanics and the keyboard layout takes a significant amount of time, but this knowledge can then be reutilized on a mobile device. In modern Western societies this learning usually occurs early in the lifetime of a person, since the vast majority of people are exposed to desktop computing and its mechanisms at a rather young age.

A different input method might be quicker to learn than a keyboard with the Qwerty layout, but if the method is one that the users have not previously encountered and utilized, it will take significantly more time and effort to utilize such an input method. Additionally, persons usually operate under a model of cognitive economy, where learning new ideas and especially changing previous learned models is generally avoided without a sufficiently good reason.

Next to the time that it takes to learn an input method, another aspect of learning relates to the ability of the users to retain gained knowledge after a certain amount of time. Humans exhibit a complex system or learning and forgetting, with many differing prevalent theories trying to describe this set of behaviours. This discussion goes mainly outside the scope of this thesis. The amount of forgetting is influenced for instance by the task complexity and previous experience of the person. Those looking for more information about learning and forgetting can refer for instance to “Human Memory: Theory and Practice”, by Alan D. Baddeley.

However, one interesting thing to observe is that the rate of forgetting seems unusually slow for continuous perceptual motor skills (for example riding a bicycle), where persons produce an uninterrupted sequence of responses. For example, Fleishman and Parker (1962) tested this by teaching a new continuous motor task (in this study a simulated flight control task) to a test group. Re-testing this subject group after two years, there was practically no forgetting

of this task, even though the users had not used this skill in the meanwhile (Fleishman & Parker, 1962).

It could be argued that some text input methods, for instance finger typing on a physical Qwerty keyboard or writing by handwriting, can be seen as similar tasks, and therefore they would not suffer from forgetting to any major extent. However the issue is naturally far more complex: for instance memory interference, both in its proactive and retroactive forms, does apply to learning text input method: What the user previously knows can hamper learning new methods and models, and vice versa, learning new models can have a negative effect on previously learned schemes.

John Carroll and Mary Beth Rosson (1987) have published a paper where they discuss "the paradox of the active user". They argue that problems in learning new input mechanisms are not simply due to bad design, but they are caused by fundamental cognitive and motivational conflicts in people. They describe two paradoxes. The production paradox is the conflict between wanting to get a task done and learning to do so. For an experienced user the issue is whether it is worth the time to stop using relatively inefficient ways of working in order to learn more effective methods, hoping that it will be worth the time and effort spent in learning. The assimilation paradox discusses the bias that previous knowledge imposes on new learning. Basing learning on previous knowledge is helpful in many instances, but apparent similarities can also blind the learners from noticing the differences between the new and previous information. Understanding these paradoxes and ways to approach these problems can help the designer to accelerate learning for a given design solution. (Carroll & Rosson, 1987)

2.4 Interaction flow with text input

Text input in mobile devices is usually only a part of a larger task that the user is trying to perform with the device. Donald A. Norman has introduced a model of human-computer interaction, called the human action cycle. The model describes how users form goals and then develop a series of tasks to achieve those goals. Each task then can consist of multiple actions with individual steps. The main stages in this model are the goal formation stage (deciding what the user wants to do), the execution stage (translating the goal to a set of tasks

and actions and then performing these actions) and the evaluation stage (verifying that the tasks and actions done have fulfilled the goals of the user). These stages are then run through iteratively, with many cycles. (Norman, 1998)

For example, if the goal of a user is contact a friend by writing him an email message, the tasks consist of launching the email writing application, selecting the message recipient, inputting the message subject and actual body text, possibly formatting the text or adding some attachments to the message and then sending it.

This task mapping can equally be done to the case of text input through a touch screen virtual keyboard. In the task sequence, when the user reaches the task of inputting text to the device, the actions are to select the desired input area (out of possibly multiple areas), activate the input method, to input desired text with the input method (verifying that the text output is correct) and to continue with the overall task, closing the input method if necessary. Some of the actions in the input task, such as the input method launching and closing can be automated in certain designs, but they can be also left to be manually controllable by the user.

The action of inputting text with the keyboard is a highly iterative process where the user thinks of the upcoming words and characters, possibly seeks them from the keyboard, proceeds to type them and then verifies from the screen that the input turned out to be as intended, correcting errors when finding them. Iteration comes also in the form of the goals of the user possibly changing during the process. Especially when writing longer pieces of text, the user might not completely know what he is intending to write, and the thoughts become clear only after starting to write them. This creates the need occasionally to modify correctly entered text afterwards, by editing and rearranging the words and sentences. Finally, after inputting the desired text, the user wants to carry on with the overall task, leaving the text input method behind.

The Norman action cycle model includes two problem areas: the gulf of evaluation and the gulf of execution. The gulf of evaluation arises when the users' perception of the world (or of a system) does not match the "actual" state of it. Likewise the gulf of execution occurs when the user does not know which actions to take in order to reach the goals that he has in mind. (Norman, 1998)

Giving a concrete example in the terms of a text input method, a large gulf of evaluation would occur if the user would come across a touch screen design looking like a virtual keyboard, but where by tapping the keys nothing would happen (in the design input would for instance be done by gestures instead of tapping). The gulf of execution in this example would then be quite clear: the user wouldn't simply know how to input text with the method, especially if there would be no affordances or hints relating to utilizing gestures.

2.5 Stylus input versus finger input

The two current main types of touch pad and screen technologies are capacitive and resistive surfaces. A resistive touch surface works regardless over what it pressed with, i.e. any physical object: stylus, pen, stick or finger can be utilized to register a touch. A capacitive touch surface requires the use of a finger to register an event, touching with a glove or a mechanical stylus does not work. The main advantages of capacitive technology are linked to better durability and scratch resistance. Using capacitive technology on top of a display also causes less dimming of the screen then when using resistive technology. (Planar Systems, 2007) Most of the current touch screen devices operate with resistive touch screens, whereas for instance nearly all touch pads on laptop computers utilize capacitive technology.

Some touch screen devices (for instance certain Windows Tablet PC devices or Wacom pen tablets) require a specialized pen stylus for the operation. These special pen styluses, usually utilizing some kind of magnetic materials, can provide extra functionalities, such as emulations for multiple different buttons (similar to right mouse button click) and the ability to detect when the stylus is hovering over screen without actually touching it. This additional hovering information can be used for instance to provide a cursor on screen, whereas with a normal touch screen contact with the screen is usually registered directly as a click event. As a drawback, these kinds of advanced touch screens can then only work with their intended pen styluses; if the intended stylus is lost, the touch screen is unusable.

There are many design differences between stylus and finger use on a touch screen. A stylus is a more accurate pointer device than the finger. For longer periods of use, using a stylus on a touch screen is less punishing than performing the same actions by directly pressing with the finger. Tapping the touch screen with the finger becomes stressful for the fingertips, since

the touch screen surface is usually very firm – consider the effects of tapping a finger against a table or a wall surface for a longer period of time. A stylus object transfers the force upwards rather than directly inside the fingers. Using a stylus even helps to keep the touch screen clean of fingerprints.

However, there are also many negative impacts to stylus usage. Stylus usage requires both hands from the user, one to hold the stylus, another to hold the device. Mobile devices are generally used periodically, for short lengths of time, and the stylus is not held when starting to use the device. The stylus is a small external object, which needs to be stored somewhere when not used. Being a small object, it is also easily lost or mislocated.

Starting to use a mobile device which requires the use of the stylus requires the user to take it out from the device and to reinsert it back to the device after the intended task has been performed. Since mobile device usage nearly always starts without the stylus, avoiding taking out the stylus completely would be highly advantageous in simplifying and streamlining their overall usage. Additionally, there is a certain indefinable 'geek' social factor currently still present with using mobile devices with a stylus; this factor starts to become highly problematic when attempting to design mass market devices.

The accuracy of a finger as a pointing device is substantially worse than a stylus, basically requiring twice as large targets in both dimensions to reach the same level of precision (see chapter 2.5.2 for details). Touch screen operating system usually register only one particular point (pixel coordinate value) as being the current touch location. This limitation usually comes from the assumptions and implementations made for desktop environments, where only one mouse cursor is assumed to be present. Pressing the touch screen with the flesh part of a finger can result in the system detecting the precise press location being in any of area of the finger that touches the screen, not only the middle point of the finger. The users' perception of the middle point of their finger might not even match the actual middle point, since this level of accuracy is not usually required in their daily lives.

However, using a mobile device touch screen with the finger has also many major advantages. Compared to a stylus, a finger is usually immediately available for usage, providing the user is not wearing gloves or having some other impediment. Taking the stylus out and putting it back in is a complicated and cumbersome activity, which does not further the actual task that

the user is intending to perform with the device. Pointing with a finger is a very primitive and therefore natural human activity. Additionally, on suitably designed mobile devices, finger use of the device can be one-handed, since the pointing hand can also hold the device at the same time in its palm.

The possibility of one-hand use is a highly beneficial feature for certain type of devices, for instance mobile phones, since then the user can use the other hand for other activities at the same time, such as carrying or operating something. This advantage does not naturally apply to all finger usable touch screen devices. Many finger usable touch screen devices and interfaces are designed, either deliberately or unintentionally, in such a manner that they require the use of two hands. In these cases in essence the finger acts only as a virtual stylus, and a potential benefit is lost.

Considering the narrower task of text input through a touch screen, the same general advantages do also apply: achieving text input without a stylus is a major benefit. However, the value of this benefit depends highly on the design of the whole system. If some other parts of the user interface in a particular task flow require the use of a stylus for successful operation, providing a finger usable input method isn't very critical or necessary, since the user is most likely already holding the stylus when the actual text input task starts.

2.5.1 Target sizes for stylus input

Mobile devices have only limited screen space available to display the various controls and information needed to manage their usage. When considering touch screen text input methods, that screen area can be used only partially for the input method, since also the outputted text, as well as preferably the context to where text is being inputted should be displayed at the same time. Touch screen text input methods which require little or no screen space screen do also exist, for instance with certain handwriting recognition methods, but a virtual keyboard is not one of such methods (unless toying with ideas relating to user interface transparency). A compromise therefore must be made, and for stylus input the question then rises of how large the individual controls of the virtual keyboard must be to be properly usable.

Defining "properly usable" isn't a trivial question. Various metrics for the definition include the speed of text input that is achievable, the rate of text input errors and the subjective comfort of the users of using such a design causes. Users can adapt to almost any size of the keyboard, but at the expense of the previous metrics. Even a tiny virtual keyboard can work in principle, but with such a design users need to write with great precision, making text input slower and mentally tiresome.

An older study from Sears et al. (1993) tested four virtual keyboard button sizes on a touch-screen PC monitor: 5.7, 7.6, 11.4 and 22.7 millimetres. They found that the input speeds were increased when the button sizes increased, in the test group of the experienced users from 21.1 wpm on the smallest keyboard to 32.5 wpm on the largest keyboard. (Sears et al, 1993)

Another more recent study from MacKenzie and Zhang (2001) tested two virtual keyboard button sizes for stylus use, 6.4 millimetres and 10.0 millimetres. In this study no significant differences between the input speeds of these two sizes were found. Error rates were slightly higher for the smaller keyboard version. (MacKenzie & Zhang, 2001b)

Sears and Zha (2003) have tested three different virtual keyboard button sizes for stylus use, ranging between 2.6 and 4.4 millimetres. Their tests also showed that there are no significant differences in text input speeds among the different button sizes. They found no significant differences in the error rates or user preferences for these three keyboard button sizes. (Sears & Zha, 2003)

Coming across studies where there appears to be no correlation between the keyboard button size and the input speed may seem surprising, but they fit the Fitts' law models (see Chapter 2.7 for description of Fitts' law).

In Fitts' law, the $\left(\frac{D}{W}+1\right)$ part of the equation basically predicts if keyboard button size is changed uniformly, then both the distances D and target widths W scale proportionally, and therefore the total value in the equation remains the same. (Parhi et al, 2006)

The differences in the test results can perhaps be explained by the nature of the tests: device use and physical movements with a physically mounted large screen monitor device (as used

by Sears in the 1993 study) may be different than when testing with a portable hand-held small screen device.

Fitts' law cannot be used as the only and definitive rule for all measurement related calculations. For instance, an additional real-life constraint for stylus based touch screen input arises from full hand movements. With a suitably small virtual keyboard the user can perform input by only moving the stylus or pointer in his fingers, without having to move the entire hand. When the touch screen keyboard total width exceeds a certain value in centimetres – depending on the size of the user's hand – accessing all buttons on the virtual keyboard becomes impossible without also moving the entire hand at the same time. This hand movement creates an additional step which the Fitts' law does not take into consideration, neither in terms of input speed or of subjective comfort. Still, in general Fitts' law provides useful and empirically proven estimates, and it can be used as a useful design principle when designing touch screen user interfaces.

2.5.2 Target sizes for finger input

In one study Pekka Parhi et al. studied various target sizes for one-handed thumb usage. They conclude that input speed increase and error rates decrease when going towards larger control sizes, but this effect stops being significant after reaching 9.6 millimetres (or higher) in touch button control size. They studied both single-target (discrete) and multi-target (serial) tasks, but the overall results for both cases are very similar. As an example, in the discrete task for a target size of 9.6 millimetres the error rate is under 3%, whereas for a target size of 5.8 millimetres the error rate raises to nearly 13%. (Parhi, 2006)

A common design rule is to aim for less than 5% error rates (over 95% success rate) for any text input method. This gives us a rule of thumb (pun intended) of approximately 1 centimetre per size of individual control in both width and height, if the controls are intended to be properly usable with the fingers. When considering that most mobile devices have a touch screen whose size is measured in single digit centimetres, the problems of designing a thumb usable input method become quickly apparent. Other fingers than the thumb, such as the forefinger, provide slightly, but not significantly, better accuracy.

2.6 Common western keyboard layouts

The Qwerty layout, named after the labels of the first six keys on top row of the layout, is the most used keyboard layout currently available (for western languages).

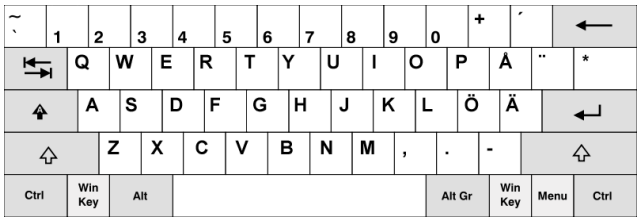


Figure 2 – QWERTY keyboard, Finnish layout (Wikimedia Commons, 2007)

Qwerty layout was designed by the inventor C. L. Sholes in the 1860's, as a layout for one of the first typewriters available on the commercial markets for ordinary consumers. There are small regional differences in the Qwerty layouts used in various countries, but they all follow principally the same basic layout.

It is generally misinformed to claim that Qwerty was intentionally designed to be a slow keyboard layout. The Qwerty layout was designed for writing the English language with mechanical typewriters. These early typewriters had a problem where the individual keys used to punch out the letters could easily become stuck if two (horizontally) adjacent letters would be pressed virtually at the same time. The Qwerty layout tries to minimize these occurrences, by not placing frequently used combinations of characters next to each other. (Gingrande, 2003)

Avoiding placing frequently used combinations next to each other is not a limiting factor considering the text input speed achievable with such a layout. Touch typing is taught so that eight fingers should be utilized to press the character keys (and the thumbs should operate the space bar). For touch typists pressing two adjacent letters isn't the quickest form of input, since multiple fingers allow almost instantaneous access equally to all parts of the keyboard. There is really no significant difference in speed over which individual finger is utilized.

Text input speeds are hampered only when the user is not able to properly alternate between the fingers of the left and right hand, and especially when the user has to type two or more

characters with the same finger (touch typing is taught so that each key is always assigned to one particular finger) (Liebowitz & Margolis, 1990). When Sholes originally designed the layout so that the use of adjacent keys would be minimized, because of these mechanical problems, he also – perhaps unwittingly – created a fairly efficiently alternating layout.

Experienced touch typists move their fingers ahead of time, anticipating future input and priming their fingers ready, so that text output usually occurs as short bursts. With typing multiple sequential characters with the same finger, the input speed is temporarily limited to how fast the touch typist can move this one finger between the various characters assigned for that finger (for example, in the Qwerty layout, the characters “d”, “e” and “c” are written with the same finger).

In a theoretically optimal layout the most commonly used characters would be located in the position where the user rests the fingers the touch typing baseline position, the left and the right hands would have approximately the same amount of use, and the use between the left and the right hand fingers should alternate as much as possible. None of these goals are fully achieved in the Qwerty layout. In practice these non-optimal issues require Qwerty users to move their fingers more than with an optimal layout, and the right hand is somewhat underutilized, but these issues do not result in slowing down the achievable input speed by any significant amount. (Liebowitz & Margolis, 1990)

It is worthwhile to note that the previous arguments countering the ineffectiveness apply at their strongest when talking about full touch typing with two hands on a full size physical Qwerty keyboard. A miniature Qwerty layout also does not usually allow the use of all of the fingers: most commonly they are usable with only one or two fingers. When typing with only one finger (or with the aid of the stylus), the Qwerty layout does create much more severe performance penalties, since the suboptimal nature of the layout can not be compensated by preparing multiple fingers ahead of time. (See chapter 2.11 for further discussion about performance optimization.)

Dvorak is the most frequently cited alternative layout to the Qwerty layout. The Dvorak keyboard layout was designed to optimize the previously mentioned inefficiencies of the Qwerty layout.

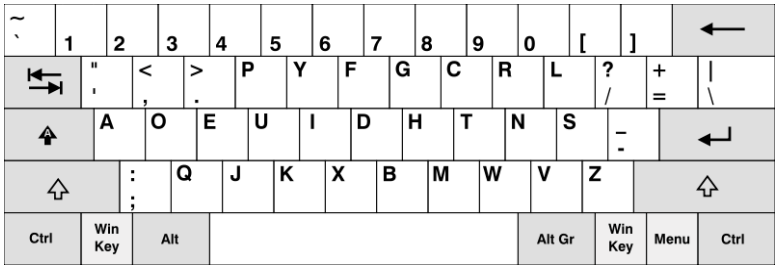


Figure 3 - DVORAK keyboard, English layout (Wikimedia Commons, 2007)

In the Dvorak layout, the eight most commonly used characters ("aoeuhtns") in the English language are located in the position where a touch typing user rests the fingertips: four vowels on the left side, four consonants on the right side (to maximise alternation between the left and the right hand). Total frequency of the usage of keys is also split almost evenly between the left and the right hand. (Liebowitz, 1990) Using optimizing principles similar to creating the English Dvorak layout, differing layouts have also been created for many other languages.

Despite the theoretical advantages, the Dvorak layout has not gained widespread popularity as a keyboard layout. The main reasons probably is – as discussed in Chapter 5.6 – the simple fact that Qwerty already is in such a large number of devices and that it works reasonably well. If one considers text input in a real life daily practice, only a portion of the total time is spent actually typing the text. Being able to type faster does not necessarily help a person to think faster, and with Qwerty experienced users can already achieve a satisfactory text input speed. The other key problem is that people cannot learn Dvorak without then hampering their knowledge of Qwerty.

In general, there isn't a vast amount of scientific study results that would indicate that the Dvorak keyboard would be a significantly faster layout. This lack of significant input speed gain is caused by many different factors, but the fundamental reason is that Qwerty is already a relatively optimized layout. As discussed previously, in proper touch typing all ten fingers are utilized, and therefore the only cases where input is substantially slower is when two or more different characters need to be inputted with the same finger. Considering input speeds it makes little difference what the actual keyboard layout is, since advanced users will learn to move their fingers beforehand in preparation of the word that will be inputted. (Noyes, 1998) As another general comment, humans have proven themselves to be very adept in

accommodating to "suboptimal" designs - many popular musical instruments serve as good examples of this. Touch typing is also a very complex and parallel process, difficult to reliably model and predict by scientific models.

However, a real significant advantage of the Dvorak layout is a reduction in the amount of hand and finger movement that is required, because of the optimization of having the most frequently used characters on the baseline of the layout. For instance users that suffer from carpal tunnel syndrome might have good justification in taking the time to learn the Dvorak keyboard layout, since it requires far less movement of the wrists and fingers in the vertical direction of the keyboard.

Next to Qwerty and Dvorak, the Abcdef layout – having keys in alphabetical order, starting from the upper left corner – is occasionally proposed and introduced in certain devices. On theory the alphabetical ordering helps initial learning to a completely novice user. However, as discussed in Chapter 2.3, the initial advantage would only last for a very short period, and afterwards the problems caused by a non-standard layout massively override the short-term potential gains. The conflicts caused by Qwerty and Dvorak apply equally well to Qwerty and Abcdef.

2.7 Fitts' law

Fitts' law, published by Paul Fitts, a psychologist at Ohio State university, in 1954 is one of the most quoted principles in relation to user interface design. It provides a model for predicting how much time does a certain rapid aimed movement take from a person.

A common formulation of this law, from I. Scott MacKenzie (Mackenzie & Buxton, 1993), goes as follows:

$$MT = a + b\log_2\left(\frac{D}{W} + 1\right)$$

where

- MT = movement time, the average time to complete the operation

- *a* and *b* are constants, determinable by doing empirical testing
- *D* is the distance from the start point of the movement to the centre of the target
- *W* is the width of the target, on the axis of motion from the start point to the target point (how wide the target is, or how closely the target needs to be hit).

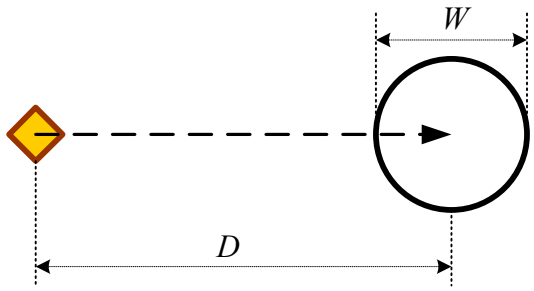


Figure 4 – Fitts' law illustrated

Essentially, Fitts' law predicts that the further away the target is and the smaller the target is, the longer it takes for the user to acquire it. Fitts' law has been proven to be accurate in many different applications, not only limited to user interface design.

To give a practical example for the values in the equation, in one empirical test done by Zhai, Sue and Accot, the value for constant *a* was found to be 0.083 seconds and for constant *b* 0.127 seconds (Zhai, Sue & Accot, 2002). This particular test was done with a virtual keyboard setup, using a stylus and resting the heel of the hand on the tablet edge. The values for *a* and *b* can vary significantly, depending on what is the pointer device, the physical configuration of the environment of the setup and the target device.

The original formula of Fitts' law has no special emphasis on the directional movement, it is effectively 1-dimensional in its viewpoint. Additionally, the definition of width in the equation doesn't take into account the various shape possibilities of the target.

In many real-life touch screen use cases, including use cases with a virtual keyboard, the hand movement occurs in two dimensions (horizontal and vertical), and the targets are not always 1-dimensional (circular). A common use case is selecting a text link on web page. This is an example of a rectangular target, such links have frequently much more width than height.

MacKenzie and Buxton (1992) have studied the effects that these variables cause. The illustration (in Figure 5) gives examples of the target object being of various shapes and in various directions.

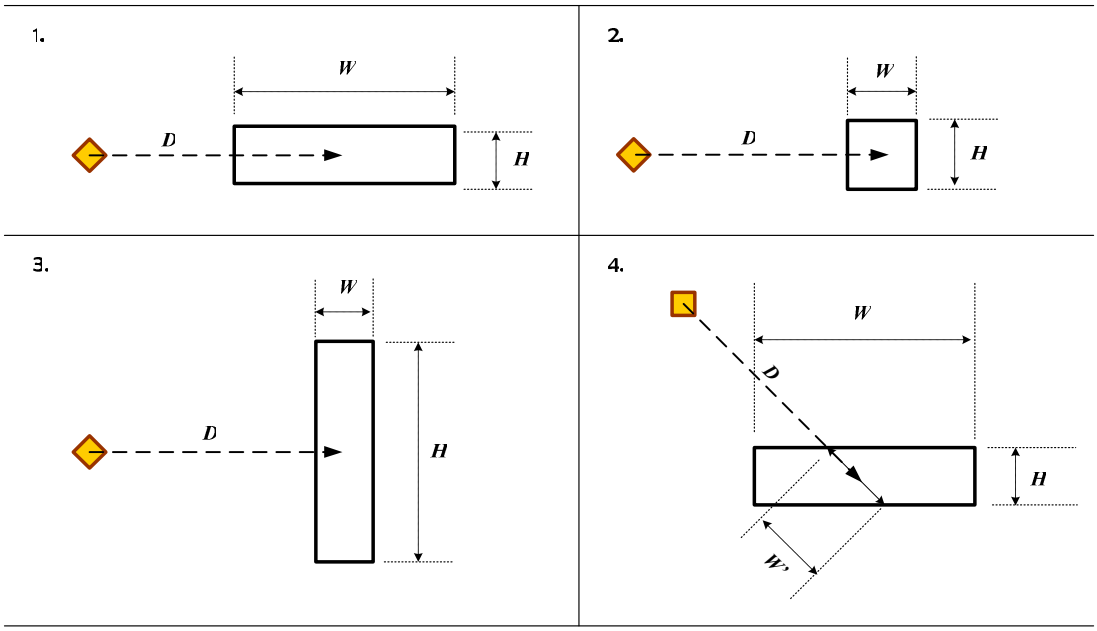


Figure 5 - Fitts' law in two dimensions, studies by MacKenzie & Buxton

Their conclusion is that whereas in the original Fitts' law *W* is utilized, more accurate results can be obtained either by using *H'* (the target diameter in the axis of movement, as seen in the fourth illustration in Figure 5) or, for rectangular targets, simply by selecting the smaller value from height or width of the target and using that as the value of *W*.

They also note that although users usually head for the centre of the target, in extreme cases the Fitts' law formula also gives incorrect values because the users rather head towards the nearest point of the object, not the centre point (as illustrated in Figure 6). This can cause quite substantial changes in the results, because these changes affect both the distance (*D*) and the width (*W, or H'*) in the Fitts' law equation.

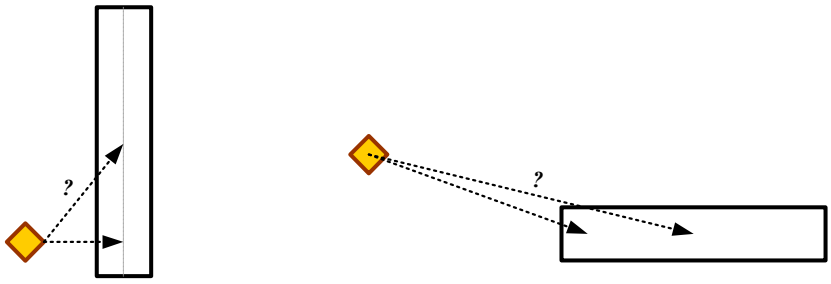


Figure 6 - Fitts' law in two dimensions, problem cases

For instance when considering virtual keyboards, usually the space bar is one such key where the users do not aim for the centre point, but rather the closest point of the target. (MacKenzie & Buxton, 1992)

Johnny Accot and Shumin Zhai have also studied the effects of two dimensional movement in relation to Fitts' law. They have refined the Fitts' law models for bivariate pointing (for cases where the target shape is not circular, but rectangular). Their mathematical formula is:

$$MT = a + b\log_2\left(\sqrt{\left(\frac{D}{W}\right)^2 + \eta\left(\frac{D}{H}\right)^2} + 1\right)$$

The formula is similar to Mackenzie's formulation of Fitts', but with the following additions:

- W is the width of the target (on the axis of motion from the start point to the target point
- H is the height of the target (perpendicular to W)
- η is a constant, determinable by doing empirical testing (approximately from 1/7 to 1/3)

In practice, this formula can be interpreted so that the height H element has 3 to 7 times less weight than the width element W – which still can be a significant effect with highly rectangular targets. (Accot & Zhai, 2003)

As a clarifying example, when looking at Figure 5, the original Fitts' law would give the same result for cases 2. and 3., whereas the bivariate version of Fitts' law predicts movement time to be faster in case 3. than in case 2, as with empirical testing it appears to be.

Grossman and Balakrishnan (2005) provide even a more comprehensive model for two-dimensional pointing, taking into account the angle of approaching the target, and allowing for non-rectangular targets. Their updated model - which is mathematically too complicated to present fully in this thesis – provides results that are mostly in line with the two-dimensional model from Accot and Zhai (Grossman & Balakrishnan, 2005).

Naturally also other target shapes than circular and rectangular can exist, but they are nowhere as common as circular or rectangular targets.

2.7.1 Mile-high menus and magic corners

In most of the commercially utilized desktop operating systems, the mouse cursor movement stops at the screen edges, even if the hand movement would result in the cursor going outside the available screen area. Because of this phenomenon, the user can access elements located on screen edges very easily, since there is no risk of overshooting the target. For example, the application menus found on MacOS applications are commonly referred as "mile-high menus" precisely because of this effect: to hit an application menu item with this design, the user only needs to worry about the horizontal aiming, vertically there is no chance of overshooting the target.

In terms of Fitts' law, in the $\left(\frac{D}{W}+1\right)$ part of the equation, in the case of screen edges, the target width W can be seen as a very large number, making the total D/W part of the equation close to minimum, irrespective of the value of D.

Screen corners have the interesting attribute of being infinitely wide both in horizontal and vertical sense. Accessing them with a mouse is very quick, since the user only needs to throw the mouse cursor in the general direction of that corner, not having to be accurate in either the horizontal or the vertical direction. Because of this easy accessibility they are often referred to as "magic corners". (Harris, 2006)

Another implication of Fitts' law is the effective nature of pop-up menus. When a pop-up menu dynamically appears right next to the current cursor position, the required movement

D for command selection is very small (or if the menus appear on top of the cursor, then nil). These principles are very useful to understand when designing pointer based user interfaces.

2.7.2 Applicability of Fitts' law with touch screen devices

Fitts' law was originally formulated to project how quickly a human could point to a physical button, but studies have shown that the same set of rules also govern how quickly a user can target an area on screen with a mouse cursor (Harris, 2006). Fitts' law has been proven to be an accurate predictive formula when using a computer mouse.

However, principles valid with a mouse are not fully applicable to a touch screen device, with stylus based interaction methods. Effects relating to mile-high menus and magic corners *can* also apply on a touch screen, but this depends on the physical properties of the device. If the user holds the stylus above screen when moving, and there is no concrete physical edge for the stylus to hit, the advantage of screen edges or the corners being "infinitely wide" is lost. In these cases the users can overshoot the target, and Fitts' law applies normally.

Even with a discernible edge the user needs to hold the stylus close enough the screen for the stylus tip to hit that edge. This requires increased accuracy from the user to hold the stylus close, but not touching, the screen when moving it, and not letting go when hitting the edge, making this kind of movement slower than when not worrying about having to keep the stylus close to the screen.

2.7.3 Exceeding Fitts' law

Fitts' law (along certain other laws, such as the Accot-Zhai steering law) is commonly used as one basis in theories that attempt to predict the text input speeds achievable by various input methods. Per-Ola Kristensson (2005) discusses that it is possible to 'break' the theoretical maximums predicted by the use of Fitts' law by utilizing predictive methods and known language characteristics (Kristensson, 2005).

For instance in the case of a virtual keyboard an additional logic can be added which predicts and automatically corrects user intended input even in the cases where the user misses some

of the correct keys on the virtual keyboard, for instance by studying nearby characters for each key press and forming the word from the most likely candidate. This in essence increases the target sizes in Fitts' law over the actual physical dimensions of the keys on the keypad, since successful input will occur even when tapping some of the nearby characters. (Kristensson & Zhai, 2005) Then again, the additional step of predictive input verification needs to be performed by the user, as described in Chapter 2.9.

These predictive methods are generally based on the assumption that the user intends to input text which is part of a lexicon of words in a particular language. This is usually true, but especially with mobile devices text input often contains words and characters that are not a part of the standard dictionary of a language. Common examples of these include email addresses, user names, passwords, web site addresses, phone numbers and chat messages. In this case the predictive methods can make text input slower than using no such methods at all.

Another method of circumventing the restrictions of Fitts' law is described by McGuffin and Balakrishnan: they describe a scheme of using user interface targets that expand in size once the user gets near to them with the mouse cursor. This in ideal circumstances leads to almost the same values as if the target would have originally been of the expanded size, although some problematic issues, for instance with target overlapping, occur with this design (McGuffin & Balakrishnan, 2005). This method as such does not apply to a normal touch screen, since it requires the use of a continuously moving mouse cursor, so that the elements can start reaching during the movement of the pointer.

2.8 Input method appraisal metrics

Various criteria exist for appraising the different mobile input methods. A common metric used is the words-per-minute (WPM) figure, which describes how many words a user is able to input with the given input method. In these figures, a word is commonly defined to contain an average of 5 characters. Since the WPM figure naturally varies for each user, based on their skills and experience, the two main strategies for determining WPM figures is either averaging the results of many individual test users, or then determining the maximum WPM figures using a theoretical model.

For example Miika Silfverberg et al. have published a model for predicting text input rates for a mobile phone keypad, basing their models on the usage of Fitts' law and on knowledge of the linguistic properties of the language in question (Silfverberg, MacKenzie & Korhonen, 2000). Another model to provide similar estimates is the combined model for text entry rate development (Isokoski & MacKenzie, 2003), generalizing to various kinds of keyboard layouts.

Related to learning models, as discussed in Chapter 2.3, another way that input methods can be appraised is how quickly they can be learned and how well they are retained in memory. Input methods are also appraised by their error rates (percentage of incorrect input). One appraisal metric is determining the Cost per Correction (CPC) average value in a given input method. The CPC value is defined as the amount of key presses that one error correction takes from the user. The CPC value can be substantially higher in the case of a predictive input method, since the user might not notice an error before the whole word is completed, or before they have scanned through all the (incorrect) alternatives that the predictive engine is offering to the user. (Gong & Tarasewich, 2005) Defining error rates is basically done by testing the input methods in practice with real users. The percentage of errors is not a fixed figure, since users can adjust their input accuracy and cognitive effort to increase or decrease the error rates to some extent.

Another useful metric for appraising is the keystrokes per character (KSPC) figure. KSPC is the number on keystrokes, on average, required to produce each character. On a full sized QWERTY keyboard KSPC is pretty close to 1.0. (Some special characters require the user to press the accent button and the character button separately.) As comparison, the multi-tap input method has a KSPC of ~2.0 – which is relatively easy to understand, since each button in a multi-tap layout usually contains three characters, and all of the characters more or less are utilized when inputting text. (MacKenzie et al, 2001)

The T9 input method has a KSPC of somewhat over 1 (usual estimates are in the range of 1.2-1.3): even though each letter is inputted with only one key press, extra key presses are created by cycling through available candidates as well as when inputting words that the input method dictionary does not yet contain. There is no exact figure for T9: the value depends on the amount of non-dictionary words that the user inputs. The figure grows slightly closer to 1 after prolonged use, since the input method design learns the custom

words and word usage frequencies of the user. Technical restrictions of mobile devices often limit the size of the dictionary and the amount of custom words the device is able to learn, making the actual behaviour not match the theoretically optimal.

It is worthwhile to note that the type and style of text that is being inputted on a mobile device deviates in the usage frequency from the norm of all text that is utilized in either speech or normal writing in a particular language. One example of this deviation is the fact that a major use case for a mobile device includes mobile messaging input, usually in the form of SMS or email messages. The language in such messages often contains unusual abbreviations, slang or combinations of characters and numbers to form words. This leads to the letter usage frequencies differing from for the language standard frequencies of a large body of published text or natural speech. (Gong & Tarasewich, 2005; Ryu & Cruz, 2005)

Additional appraisal metrics can be formed for instance by measuring subjective opinions of comfort, stressfulness or the amount of mental effort required with a certain method. Most appraisals produced by these kinds of methods are then qualitative instead of quantitative, but the data can be much more revealing and usable than a quantitative number of quality given to a certain input method. Many of the methods for qualitative usability studies, such as a questionnaires, interviews and open discussions are readily applicable for text input method studies.

2.9 Predictive text input methods

Non-predictive methods, such as a normal physical keyboard, require the user to input each character exactly and non-ambiguously, by explicitly selecting or entering each character. However, human languages are not random. A language lexicon defines most character combinations to be meaningless gibberish, and language grammar defines most word combinations to be incorrect. Predictive methods utilize these rules by predicting what the user intends to input, or what the user has already inputted, from an ambiguous set of input signals.

Prediction can be seen to occur in two fronts: in a narrow sense prediction is about what the user is intending to write. Methods such as word completion try to predict the future input

from what the user has already written previously. In a broader sense predictive methods are all text input methods where text is deciphered from an ambiguous set of inputs from the user (MacKenzie & Tanaka-Ishii, 2007). By this broader definition also for instance handwriting recognition can be seen as a predictive input method.

One of the most well known mobile predictive – here seen in the broad sense – input methods is the T9 input method from Tegic. On mobile phones T9 is usually implemented on top of the ITU-T keyboard layout. In the T9 model, a word is inputted by pressing each key that features the intended character once. The predictive system then calculates the possible words and presents the most likely candidates to the user. (Tegic Communications, 2006) Common variations of the T9 method include LetterEase and WordWise from Eatoni Ergonomics (Eatoni Ergonomics, 2007), and the SureType input method from Research in Motion (Segan, 2007).

To give a practical example of the T9 predictive method, looking at Figure 1 on page 9, in the ITU-T layout button “8” contains the characters “tuv” and button “6” the characters “mno”. When the user presses the buttons 8 and 6, the first logical predicted candidate is the word “to”, whereas for instance “tm” is not a valid word in English (unless the user trains the system to accept “tm” as a valid word).

A fundamental trait of predictive methods is the increased need of user interaction and feedback. Whereas with non-predictive methods the user can be sure of what text will be inputted and can continue to input without any interruptions, when utilizing any predictive method the user constantly needs to exhibit an additional step of verifying that the prediction provided in the system turned out to match what the user intended. This extra step of verification is the critical difference in relation to non-predictive input methods, and it simply cannot be totally removed. Basic verification is also occasionally needed with non-predictive methods to check for typos or spelling errors, but it occurs when the user wishes to do so, not when the design forces the user to it.

The extra step takes time and mental effort from the user, and it constantly breaks the basic flow of text input. Desktop devices do not usually have predictive text input systems, arguably because of these reasons: the additional step of verification takes more time and mental effort than what is saved by utilizing the prediction features. Desktop devices already

provide a full size hardware keyboard, so one can argue that there is no need for prediction in the first place. For mobile devices, the constraints in terms of the physical properties of the device are much more intense.

The intensity of the cognitive challenge differs between various predictive input methods. For example the T9 input method exhibits the behaviour where the currently displayed predicted candidate might not match the final input. A word candidate is shown, but the user inputs the next character, the predicted word might change to a completely different word. This design also creates what can be known as error amplification, where with only one incorrect button press the whole word will turn out to be incorrectly predicted, without the user possibly not noticing the incorrect key press, since it is extremely hard for the user to notice this error, due to the constantly changing nature of the predicted word. (Kober et al, 2001)

The LetterEase input method tries to improve upon this error amplification phenomenon by making only the last inputted character predictive, i.e. input of following characters will not change the previously inputted text. On the other hand, this essentially means that the users need to verify input after selection of every character, whereas many users would prefer to write in blocks or bursts of a couple characters at a time, and only then verify that the input is as desired. (Ryu, 2005)

The general usefulness of predictive input methods naturally varies. For instance T9 has proved itself to be a significantly faster text input method than the ITU-T multitap input method with experienced users, although the mental effort it requires makes it unpopular with a percentage of users. In one test by James and Reischel, experienced users could input text nearly 20 words per minute with the T9 input method, whereas the same users could reach a speed of 8 words per minute using the multitap input method (James & Reischel, 2001). These speeds are substantially less than what can be achieved with a mobile device. In a test by Clarkson et al. (2005), testing two miniature hardware Qwerty layout keyboards, novice users achieved approximately 30 words per minute, and expect users approximately 60 words per minute with these keyboards (Clarkson, 2005).

On the other hand, having a predictive feature such as word completion is not always so clearly beneficial. For instance, in one study Költringer and Grechenig tested the

WordComplete 3.0 word completion software alongside the PalmOS 5.2 virtual keyboard. They found no significant impact to input speed or error rates. The users did comment that the subjective task load was lower when utilizing the word completion functionality. (Költringer & Grechenig, 2004)

The results of these word completion studies are relatively understandable. Scanning the word completion candidates takes time, and the desired word is not always displayed or found among the candidates even after scanning. This scanning decreases the efficiency so that a similar amount of input could be performed by simply writing the intended words fully. However, input with a stylus on a virtual keyboard can subjectively feel stressful and cumbersome. The reduction in the total amount of taps required, and the general positive feeling generated by a "smart" feature such as word completion can increase the subjective opinion towards such a feature, even though it wouldn't make the actual text input task any faster.

2.10 Assistive methods for touch screen text input

Various methods of assisting the user with the task of touch screen text input have been created. One of the most frequently provided assistive methods is auto-capitalization. Since normal text input requires the user to switch between the upper and lower case keys (to start a sentence with an upper case character), in the auto-capitalization feature the keyboard layout is automatically switched to upper case layout when the system determines that the user is starting a new sentence. Auto-capitalization is not always desirable, for example when inputting case-sensitive information, such as passwords.

Another method of assistance is to provide visual cues on the virtual keyboard. Laurent Magnien describes a method where, based on language patterns, the most likely keys to input the next character with are visually highlighted from the keyboard. This would be of use especially to novice users who do not yet fully know the keyboard layout. (Magnien et al., 2003) A similar method of providing visual cues about the most likely next characters has also been studied for a predictive multi-tap layout (Gong, Haggerty & Tarasewich, 2005).

Figure 7 – Example of providing visual cues for input assistance (Magnien, 2003)

Another category of assistive methods are those that provide automatic error correction for the user. The system can contain information about common misspellings, character reversals inside a word, accidental punctuation, and missed or double-tapped keys, and correct them automatically. Another variation of automatic correction relates to internationalization and the input of accented characters: in certain designs, the system can automatically add correct accents to characters. For instance, in Spanish if the user would type "manana", the system would convert the input to "mañana", saving the need to input the accented character manually.

An alternative to automatically correcting words is to highlight the suspect words on the screen and provide alternative candidates for them upon request, so that the users can review for themselves whether they need to be corrected or not, and select the correct candidate if an error is found. For example the recent versions of the Microsoft Word text editing application provide assistance similar to this to their users.

Another category of assistive methods, related to automatic error correction, is the use of customized or abbreviated input. The difference to error correction is that here the users intentionally write the content that gets corrected. The system can convert abbreviations and agreed shortcut phrases to words and sentences, for example writing "CUL8" gets converted to "See you later".

The effectiveness of these automatic methods depends heavily on the quality of the language lexicon in use. In general, all methods that automatically correct the input from the user can create some unintended errors. Users do not always want to input lexically correct text: many

uses of text and language in modern messaging are based on intentional misuse of grammar, in the form of slang or abbreviation use. Much Internet based communication, such as web site addresses, user names, email addresses can feature names and words that are not part of the lexicon, but which still should not be automatically changed to their correct counterparts. In a chat discussion the user might want to actually write "CUL8", and not have that phrase converted to a grammatically valid sentence.

Next to correction methods, word completion methods predict user upcoming input from previous input. For instance, if the user inputs "see you to", the system can deduct that the user may intend to input the word "tomorrow" or "today", and provide these as word completion candidates. Completing the word then requires only one selection from the user, instead of typing every character of the word. Word completion methods can even attempt to predict multiple upcoming words at the same time. A close relative to word completion is the next word prediction, where the system attempts to predict the next upcoming word even before the user has inputted any characters of that word. In this case, the system predicts "tomorrow" as a next word candidate immediately after the user has inputted "see you".

One way to rate the usefulness of a word completion method is to consider how quickly users write text with the input method normally, how long on average does it take for the user to notice and select the correct candidate, and how many characters an average candidate contains. If it takes more time to notice and select the candidate than it takes to simply write the word in question, one can argue that the word completion feature isn't proving to be useful. Naturally the text input speed is only factor to consider – justifications for word completion can come from reducing the total number of presses, and therefore making input more comfortable and pleasurable, even if the actual process of inputting wouldn't become faster.

Analysing predictive input methods in regards to the Norman human action cycle, all ambiguous methods run the risk of creating a gulf of execution for the user, or at least creating many cycles of evaluation and execution. The user cannot always know for sure which actions he must execute to perform the desired action, in this case to input a certain phrase. Whereas with non-predictive methods the users can learn long phrases of execution, with predictive methods constant switching between execution and evaluation is required.

2.11 Input method layout performance optimization

Touch screens, when used with a stylus, involve the movement of one hand and therefore one pointer. Text input with a virtual keyboard becomes a series of discrete pointing tasks. Both Dvorak and Qwerty layouts try to achieve alternation between the left and right hand, to make two-handed keyboard usage more efficient. Considering the nature of stylus input, alternating between the two sides of the keyboard is actually undesirable. Fitts' law predicts that faster input would occur if the frequently used keys would be as close to each other as possible, not if they require constantly moving between both sides of the layout.

Since touch screens do not share the same physical constraints as a real-life keyboard, efforts can be made to further optimize the keyboard layout, also in relationship to the amount, sizes and positions of the character keys. Even the shape of the overall keyboard can be altered: for example, to minimize the amount of required hand movement, a square keyboard would be a more effective shape than a rectangle.

On a theoretical level, performance optimization can be done by calculating the transitional frequencies from one character to another in a given language. Ideal performance would be achieved by laying out the characters so that the total travel distance of the pointer over a lengthy text would be as short as possible. In a virtual keyboard this can be achieved by designing the layout so that the most frequent keys are located in the centre of the keyboard, and by having the frequently connected letters (digraphs) closer to each other than the less frequently connected letters.

Zhai describes the use of the Metropolis random walk algorithm in designing one theoretically optimized keyboard layout. The result of his work is the Metropolis keyboard.

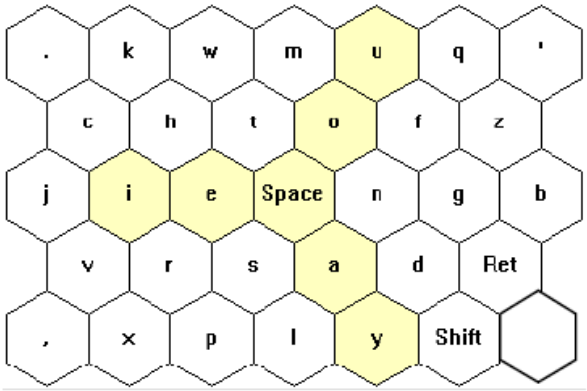


Figure 8 - The Metropolis keyboard (Zhai, 2000)

The theoretical maximum input rate of the Metropolis keyboard is 43.1 words per minute, which is over 40% higher than with the same calculations done to a normal QWERTY virtual keyboard layout. (Zhai et al., 2000)

Smith and Zhai further refined this design principle by adding an alphabetical preference to the algorithm, so that letters early in the alphabet are more likely to be at the start (the top left side) of this layout. This brings no theoretical advantage, but it makes the overall layout easier to learn. Testing the difference between a random order keyboard layout and one with alphabetic preference, the real-life input speeds were increased by 9% in the alphabetically optimize version, and the learning of that layout proved to be quicker. (Smith & Zhai, 2001) The input speeds and the learning speeds are here naturally closely connected to each other. One can speculate that with sufficient time the differences in the input speeds would start to disappear.

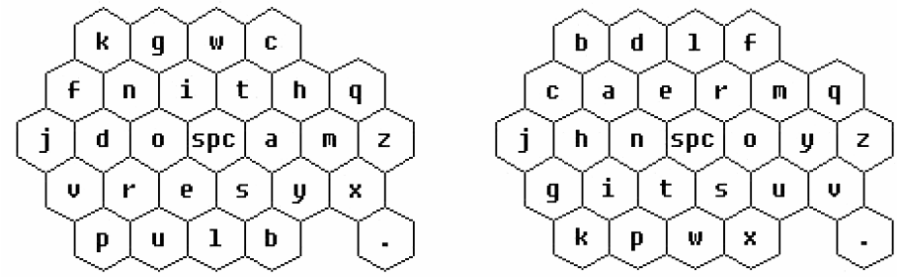


Figure 9 - Random placement vs. alphabetic preference on an optimized keyboard (Smith, 2001)

Other common examples of performance optimized keyboards include the Atomik layout, the Fitaly layout and Opti/Opti II layouts (MacKenzie & Zhang, 1999). The details and character placements differ, but they all follow the same basic principles, trying to optimize for speed by reducing the required amount of traversal between individual keys.

The end results of these studies are keyboard layouts that are significantly faster to input with than with desktop Qwerty or Dvorak layouts, after a significant amount of user learning and training. Whereas with Qwerty and Dvorak the risk of conflicting skill transference is very high (i.e. learning Dvorak hampers the Qwerty skill of the user), the more an optimized layout does not resemble the traditional Qwerty layout, the less problems with memory interference learning such a layout and method would cause for the user.

2.12 Touch screen input versus hard key input

So what are the main differences between hardware and virtual keyboard keys? Why are virtual keyboard input solutions unable to compete with proper hardware key solutions?

A critical advantage of hardware buttons comes from their haptic nature: users can find and locate the keys by touch. Additionally, the users can also place their fingers beforehand on top of the keys without causing input to occur, allowing input in short effective bursts. Furthermore, hardware button gives concrete physical feedback when it has been sufficiently and successfully pressed. Multiple hardware buttons can be pressed at the same time (for instance the Shift key and a character key at the same time) with ease. Dual touch and multitouch touch screen devices are under study, and eventually coming to the market, but they are not yet readily available.

With these advantageous properties, an experienced user can input text with a hardware keyboard without having to look at the keyboard at all, operating solely on muscle memory and on the haptic feedback provided by the keys. This is virtually impossible with a touch screen solution. The feedback element can be emulated to some extent in certain touch screen devices, for instance by adding piezoelectrical or vibrational hardware components to provide haptic feedback. Touch screen are unable, at least with the current technology, to

dynamically provide physically raised surfaces for locating the keys. The user is usually also unable to rest his fingers on top of the keys without causing activation.

Critical to the performance, with hard key solutions the users can use multiple fingers simultaneously and comfortably, preparing for and setting up the upcoming characters in the input sequence in a parallel manner. Touch screen input is usually done with only one (finger or stylus) pointer. In this case Fitts' law dictates the minimum times it takes to move from one character to another. Using multiple fingers with hard keys reduces the distances that need to be travelled to reach the desired keys.

As previously mentioned, fixed physical keyboard layout allows the development of muscle memory for the user. A proficient user of a physical keyboard does not need to look at the keyboard, but he can keep his gaze constantly directed towards the actual text. This reduces the need for eye movements (users are usually unable to perform any actions or note changes in user interface during the eye movement saccades) and makes immediate error detection possible (Jacob, 1995). With a virtual input method the user needs to look at the virtual keyboard when using it, therefore splitting his gaze the input method and the actual inputted text. Also, whereas with a virtual keyboard the user always needs to verify from the screen whether input turned out to be correct, a truly proficient user can simply immediately feel making a mistake, even before looking at the screen.

2.13 Advantages of virtual keys over hardware keys

As discussed in the previous chapter, even though in general a hardware key based input solution is mostly superior to a touch screen based solution, a touch screen solution can provide certain features that a hardware solution is not able to provide. The limitations and standards prevalent in physical keyboards do not need to apply to virtual keyboards. The layout, dimensions, the amount and the placement of the keys on a virtual keyboard is only limited by the device screen dimensions. Multiple different versions of the keyboard can exist in the same device, and the user can switch between these layouts. For instance rarely used characters and symbols can be provided in an alternative and customized "special characters view" of the virtual keyboard.

In theory, the layout and dimensions of a virtual keyboard layout can even adapt during the use of the virtual keyboard. Johan Himberg et al. describes a touch screen keyboard design that slowly moves the keys and adapts the keyboard layout and sizes of individual buttons to better match the physical movement patterns of the user, by recording and analyzing the button presses done by the user (Himberg et al., 2003).

On the other hand, including multiple different layouts into a text input method decreases the overall learnability of the given input method. As discussed in Chapter 2.3, a critical step in improving text input speeds is the motor reflex acquisition phase, where the user starts to learn the input method layout in a manner where there is no need to visually search the layout every time a certain character is to be inputted. The positive aspect of a fixed keyboard layout and configuration comes from its learnability: the input speed starts to increase substantially once the user learns the keyboard layout and is ensured that the layout remains the same. If the layout would constantly change, then this learning would never become possible. With multiple separate constant layouts, this learning is also possible, but it takes longer than for a single layout.

As discussed previously, the rectangular form factor of a physical Qwerty keyboard has its roots on being suitable to be used with two human hands, resting on the middle of the keyboard. A virtual keyboard is usually used with a stylus, which is similar to typing with only one finger. With these constraints the rectangular shape of the Qwerty keyboard is not ideal. Basing on Fitts' law, a square (or circular) shape would be closer to the most optimal shape.

A mobile device almost always contains only one physical keyboard, designed as a compromise to fit all the relevant text input tasks and user groups, sometimes supporting multiple different languages. Languages differ in many aspects, even in the number of characters utilized: thus for instance a mobile device hardware keyboard that has been optimized to input English will encounter difficulties with languages that have more characters than English, for example when attempting to input Finnish, with its added Scandinavian letters "ä", "ö" and "å".

A major advantage of a virtual keyboard is the ability to localize the keyboard into various language variants, enabling the use of the same device in multiple languages and regions,

even going down to the level of slang and subcultures inside one language. Different language variants can contain a different number of keys on screen, and in different positions, better fitting the characteristics of each supported language.

A virtual input surface allows for far more dynamical predictive and assistive input methods (see Chapter 2.4) than a static physical keyboard. The controls and the contents of the input method can be as context-sensitive as the design requires. The predictive capabilities can also be highly customized based on each selected language.

With virtual input methods, the users can also have the possibility to customize the input method to match their individual personal preferences. Customization can range from selecting different sizes for the input method to modifying the keys and layouts that the system presents to the user. One part of customization is also the ability to upgrade the input methods: new methods can be installed to the device and the current ones, their design and behaviour upgraded with device software updates. With a physical input method such upgrades are generally not possible or practically feasible.

To summarize, the selection of a touch screen solution over a hard key solution has many potential advantages as well as disadvantages. The various possibilities and alternatives for a touch screen design solution are considerably greater, but so is the potential complexity and difficulty in learning to use such a solution.

3. BASELINE DESIGN ANALYSIS

The second study objective of the thesis was to analyze the baseline device that was available at the start of the design project. This chapter will introduce the baseline device and its UI design for the touch screen text input methods. The strengths and weaknesses of the design will be analyzed.

3.1 Platform and device introduction

The baseline design utilized in this design project is the Nokia 7710 widescreen smartphone. Nokia 7710 is a touch screen mobile phone, utilizing the Symbian OS Series 90 operating system. Symbian OS Series 90 "-- is a complete and feature-rich development platform providing an excellent environment for media consumption, mobile services and a true Internet experience. -- The UI features interactive onscreen components, such as handwriting recognition and an Onscreen/Virtual keyboard, enabling the user to control a device with a stylus or hardware keys." (Jerz).

The Nokia 7710 is the only commercially available Series 90 phone. Nokia 7710 is the successor to the Nokia 7700 device, which never was released into open commercial sales. Development of the Series 90 software platform has since been merged into the Symbian OS Series 60, and Series 90 as a separate platform is not developed any further.

Figure 10 - Nokia 7710 Series 90 widescreen smartphone (Nokia, 2007)

The Nokia 7710 has no physical keypad for text input, all text input is performed through the touch screen. The device features a broad range of common phone, communication and PIM applications, as well as telephony and connectivity through cell phone wireless networks.

The device has a touch screen of 640x320 pixels, in landscape format (a 2:1 display aspect ratio). The physical measurements of the display are 8.2 x 4.1 centimetres.

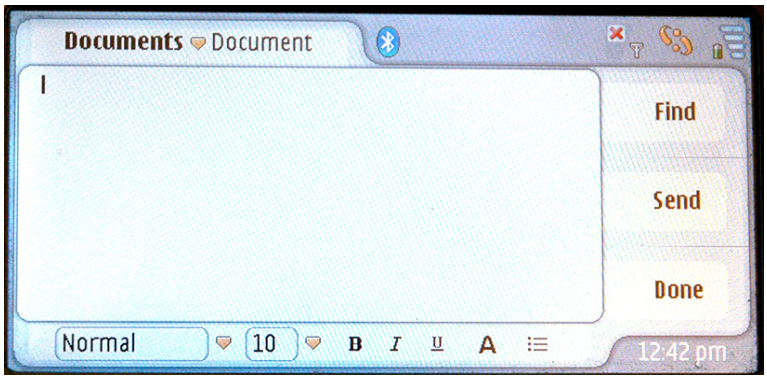


Figure 11 - 7710, sample of application layout - Documents application

The Nokia 7710 features an UI style evolved from the Symbian Series 60 and Series 80 edition phones. Many changes and alterations were made so that the device and the user interface style would provide good usability with a touch screen. Virtually all operations of the device can be performed using only the touch screen, not requiring hard key interactions.

The device supports running multiple applications simultaneously, although only one of those applications is displayed on screen at one time. The user interface model portrays applications as always running. No definite list of running applications is presented to the user (although a list of recently utilized applications is available), and running applications do not feature a command or button for explicitly closing them. The user is able to switch between applications by selecting them again from the main application menu. The management of running applications (and device memory) is left to be handled by the device and its operating system.

3.2 Analysis of baseline text input UI design

Analysis of the text input method UI of the baseline device is done partly by expert evaluation by the author, and partly by utilizing the already available user and usability test

studies and results for the product. Some of the available user and usability study result material is confidential and unfortunately cannot be fully shared in this thesis.

Note that further analysis will restrict itself to issues related to the virtual keyboard input method of the device, not to other parts of the device UI. The Nokia 7710 device also provides handwriting recognition as an alternative input method to the virtual keyboard.

3.2.1 Input method activation and closing

In the Nokia 7710 user interface there is no dedicated UI control or button to launch the virtual input method. Launching the input method window is done by tapping once with the stylus on the input field where the user wishes to input text, or by pressing the device "Select" hard key when a text input field has been focused.

Many competitive touch screen operating systems operate with a different principle: they provide an input method launching control somewhere in the UI, often in the status bar or tool bar area. Tapping this control brings the virtual input method window into view, tapping the control again hides the input method from screen.

As far as closing the input method, the Nokia 7710 the input method layout provides a button for the user to close the input method. Additionally, the input method is automatically closed when the user navigates away from the view or usage context to which he was inputting text. See Figure 13 for more details about the available controls in the input method window.

3.2.2 View layout with text input method

The Nokia 7710 text input method is presented as a floating window. When the input method is launched, it is displayed on top of any current device view or content (see Figure 12).

Figure 12 - Nokia 7710 text input method: virtual keyboard as a floating window

The floating window consists of multiple functional areas (see Figure 13).

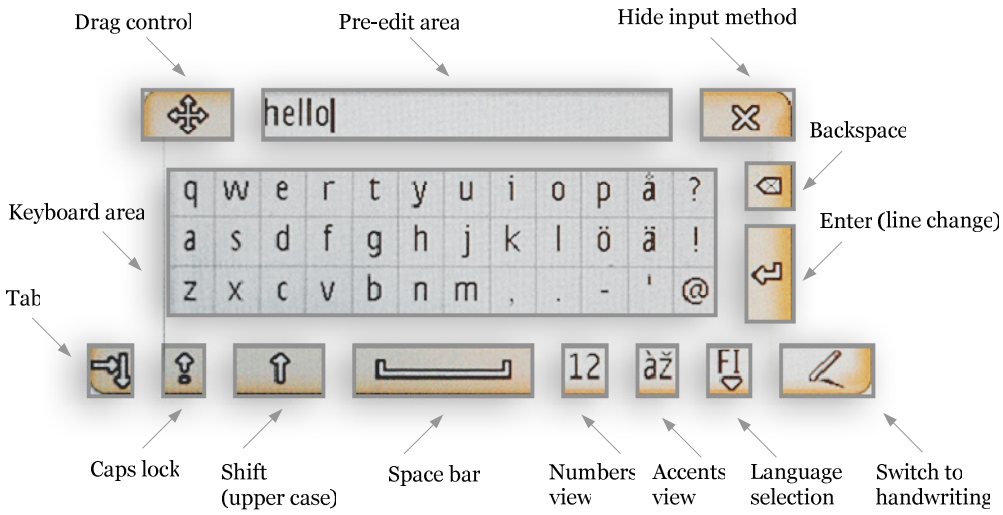


Figure 13 - Nokia 7710 virtual keyboard text input method: functional areas

Most of the controls of the virtual keyboard (backspace, line change, enter, shift, caps lock) function similarly to their physical keyboard life counterparts. The keyboard contains buttons that allow the user to switch a different layout (to accents numbers or accented characters), a different language or then switch to the handwriting recognition method.

The top part of the Nokia 7710 virtual keyboard input method contains a pre-edit field. The pre-edit field displays a line of the most recent text that the user has inputted. With this design any text that the user has inputted appears instantly in the pre-edit field as well as the actual application area. The pre-edit field allows the user to see inputted text even when the floating input method covers the actual text field where the text is being inputted. The pre-

edit field is located directly above the keyboard area, reducing the need for eye movements (see Figure 14).

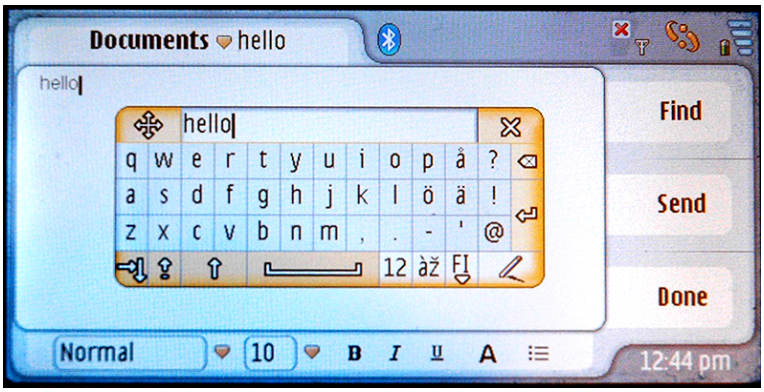
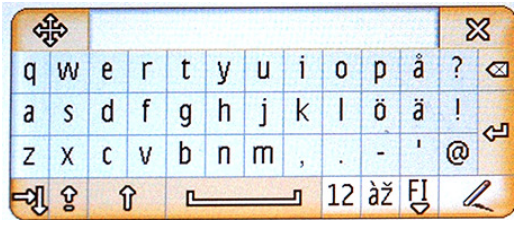


Figure 14 - Nokia 7710 text input method: input in pre-edit field and in the application

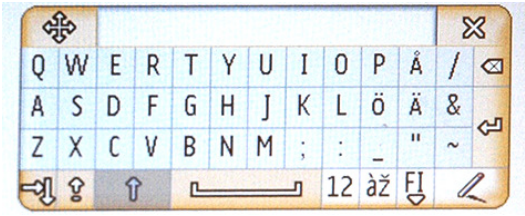
The user is able to move the floating input method on screen by dragging it from the top left corner drag control. The input method window can be dragged partially off-screen, to show more of the application area content.

3.2.3 Text input and editing layouts

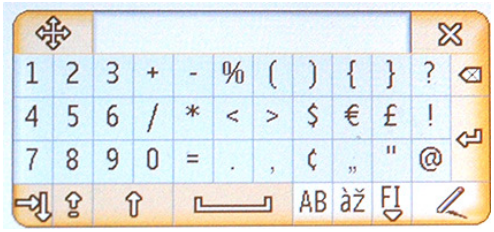
The Nokia 7710 virtual keyboard input method consists of multiple layouts: one for lower case characters, one for upper case characters, one for numbers and related symbols and two layouts for accented characters (see Figure 15).



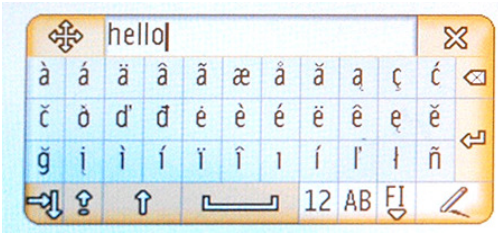
Lower case layout



Upper case layout



Numbers layout



Accented characters layout 1

Figure 15 - Nokia 7710 text input method: virtual keyboard layouts

The lower and lower case character layouts also contain some commonly used punctuation and other symbols. The total horizontal width of the input method window changes slightly when switching to the numbers and accented characters layouts, so that no unnecessary space is taken from the screen. The horizontal size also depends on the language selected, with certain languages containing more commonly inputted characters and therefore making the window width slightly larger.

3.2.4 Assistive methods

Of the assistive methods described in chapter 2.8, the Nokia 7710 device provides auto-capitalization: the layout is automatically shifted to upper case when it is required to start new sentences. Automatic mode selection is also provided to a limited extent: when the user selects to input text to a field that requests numbers, the layout is automatically selected to display the numbers layout, and unsupported characters are dimmed from the keyboard.

Word completion or other similar predictive methods are not provided in the device.

3.3 Design problem areas

The following data is gathered from summarizing the results of existing usability studies on the baseline device, as well as the author's own study and analysis. These usability studies are unfortunately not public material, but a summary of the test results can be presented here.

The UI design of the text input method in the Nokia 7710 has many aspects that fare well in the usability tests. For instance the tasks of launching the input method, basic typing, editing and layout switching have high success rates.

However, moving on to the problem areas of the design, the fundamental problem with the floating input method arises from the fact that it overlaps the application area and its content. In the worst case the floating input method overlaps the actual text input field to where the

user is inputting text. This trait of the design introduces a requirement for the user to interact manually with the view content and the input method window, moving the input method on screen so that it would not cover relevant content.

The device does have a system where it automatically attempts to open the input method to such a position on screen where it wouldn't overlap the actual text input field. But the device cannot know in all cases which parts of the content area are relevant for each user at each particular time. Even if the input window avoids overlapping the text input field, it may overlap some other critical content in the view which the user would need to see in order to know what to input.

The introduction of a pre-edit field, although trying to solve the problems created by the floating input method window, creates additional problems of its own. The contents of the pre-edit field do not always match the contents of the application area, especially right after the cursor has been re-positioned. The pre-edit field only fits a small row of text, which becomes insufficient when the user attempts to evaluate a full-length inputted sentence (forcing to move the floating input method anyway). In general, having two fields with usually mostly similar contents is easily confusing, especially in the cases where the contents do not suddenly match anymore.

In certain usability tests the users have complained that the automatic launching feature of the input method did not appear to work in a consistent manner. Sometimes the input method was already displayed when reaching a certain text input method, but sometimes not, seemingly without any greater logic. The users were especially annoyed when the input method launched automatically in situations where the users did not wish to input text. The tested implementation featured some views and tasks where the keyboard was automatically activated, without the user manually tapping it visible.

Although layout mode switching is handled fairly successfully, many users in the usability tests complained about the unfamiliarity of the presented input method layouts. Some of the layouts differ substantially from the physical Qwerty keyboards and layouts that users are accustomed to. In overall, the input method appears visually quite complicated, with a large number of different buttons and controls around the actual keyboard area. Users also had difficulty in understanding some of the icons for the functions of the virtual keyboard.

The main location for symbols and special characters in this design is the numbers layout, but also the upper and lower case characters layouts (as seen in Figure 15) contain some of the commonly used symbols and special characters. This results in the users having trouble in knowing where a particular symbol or special character is located – whether a particular symbol is in the lower case, the upper case or the numbers layout.

The different views of the device virtual keyboard were of slightly different physical sizes. This meant that switching from one view to another resulted in the entire input window repositioning on screen, harming eye tracking to a significant extent. The control which the user pressed to switch to the numbers layout also sometimes moves on screen when the layout switches. Users also had difficulty in finding some special characters, since they weren't sure in which of the various layouts the particular character would be located.

Increasing the problems, a floating input method in use can cover even more area than what the input method seems to take on the screen. The hand of the user can cover an additional extensive area of the screen, mostly below the floating input method window.

In general, the text input method design in this device serves as a useful design baseline. Many of the individual positive aspects of the UI design can be retained, while others show potential for changes and improvements.

4. DESIGN WORK

After evaluating the baseline design, this chapter will present the UI design of the touch screen virtual keyboard input method done for the Nokia 770 Internet Tablet device. Special focus is put to discuss the reasoning when deviating from the baseline design.

Naturally a large team of designers, consultants, product managers, software and usability engineers was involved in this design process, in its various stages, providing many useful ideas and arguments for the design. The author's role was to gather all this input from various parties, take them into consideration, decide upon and create the final design, driving it then forward through the organisation and the implementation process. Additionally another task was to write the actual user interface specification, in close co-operation with software implementation parties, to ensure implementation feasibility.

4.1 Platform and device introduction

The target for the updated UI design was the Nokia 770 Internet Tablet device (see Figure 16).

Figure 16 – Nokia 770 Internet Tablet (Nokia, 2007)

The Nokia 770 Internet Tablet shares certain similarities in its design and properties to the Nokia 7710, for instance both of the devices have a relatively large touch screen, and both

feature hardware buttons on the sides of the device. The key difference, apart from the different operating system, is that the Nokia 770 Internet Tablet is not a mobile phone, it has no direct cellular connections and it therefore cannot support calling to traditional cellular networks. Internet connectivity to the device is provided through wireless networks and with Bluetooth connections via mobile phones.

The Nokia 770 Internet Tablet has a landscape touch screen of 800x480 pixels, physically measuring 9.0 x 5.4 centimetres (with a 5:3 display aspect ratio).

Note that the images used in this thesis of the Nokia 770 Internet Tablet are from the 2006 software edition, which has some added features. The 2006 software edition shares the same main functionalities in the input method design, although it features minor updates to certain behaviours and layouts, and a new full screen keyboard input method (which is not described in this thesis). For the purposes of this thesis I will limit the description of the text input features to the first (2005) software edition of the product.

Figure 17 – Internet tablet, sample of application layout - Browser application

The touch screen user interface of the Nokia 770 Internet Tablet device is split into multiple areas (see Figure 18).

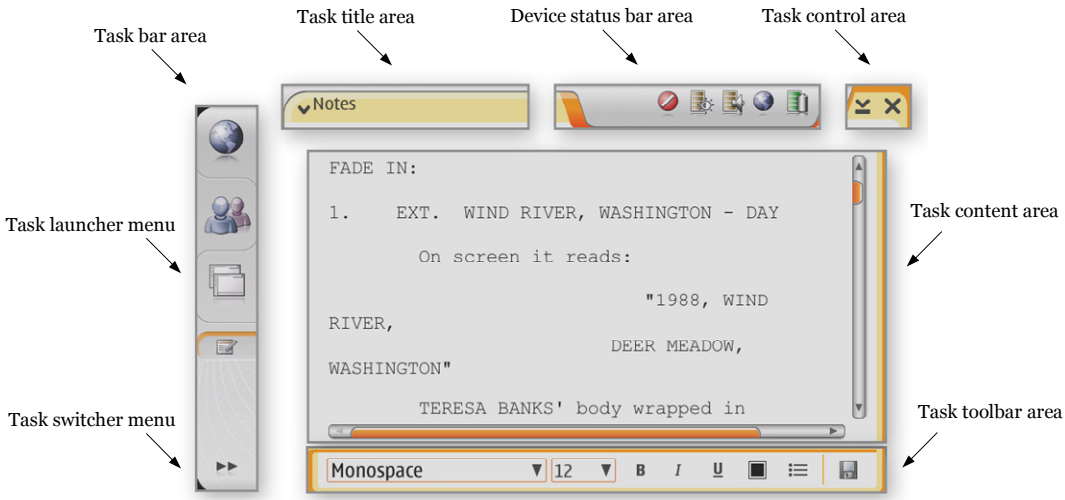


Figure 18 - Nokia 770 Internet tablet – common UI control areas

The left side of the view provides the user access to task (application) launching and switching, as well as shortcuts for the most commonly used features of the device. The top side of the screen displays the device status bar area, as well as provides information and controls for the currently running task. The main area of the view is reserved for the currently ongoing application or task. The device features an additional full screen view mode, where only the top and left control areas are hidden, and the full screen is provided for the actual task content area.

The UI allows the user to run multiple applications simultaneously, but only one of them is displayed on screen at one time. Each task is presented as a task icon on the Task bar area. Switching between running applications is performed either by tapping the task icon, or selecting the task from the Task switcher menu. As a difference to the Nokia 7710, in this device the user is able to manually manage and close running applications.

4.2 Improvements for the baseline design

The baseline design review and evaluation gave various targets for improving the design and the usability of the on-screen text input method. Changes to the baseline design were necessary also for many others reasons. For instance the physical properties of the target device were different from the baseline device. The screen size and shape was different, as

was the amount and layout of the hard keys in the device. The software platform was also completely different, which then created a new set of opportunities and challenges in implementing certain features.

Considering the analysis of the baseline design (found in chapter 3.3), the floating input method area design was seen to cause a large number of the observed usability problems. Ways to solve those problems would need to be studied, either by improving upon the aspects of the floating input method design, or then by switching to a completely different design. The need for unnecessary manual interaction (dragging the input method window or moving the content area to avoid overlapping) was seen especially troublesome.

Another aim was to study ways of bringing the virtual keyboard design closer to something that users would instantly identify as resembling a conventional physical keyboard. Related to this task, the different views of the virtual keyboard were to be streamlined and harmonized, so that switching between the views would result in minimum confusion. If possible, the number of different views was also to be reduced. Rarely used controls were not to be directly presented to the user, to reduce the amount of constantly displayed controls.

Software components acquired for the product would also allow implementing more advanced predictive and assistive methods for text input. A design that would utilize the available predictive input methods in a reasonable manner was to be introduced as a new element to the overall text input design.

Additional smaller usability improvements were also to be made if these would be seen appropriate and possible to implement. Since the Nokia 7710 device and its input method design had comprehensive usability testing data available, unnecessary deviation from that design was to be avoided, so that the design could rely on that available data where still applicable.

4.3 Text input method design

The main issues solved in the design of the new text input method will now be presented. The design utilizes both the knowledge constructed in the research study as well as knowledge of the issues raised with the baseline design. Special focus will be put to discussing

the major changes done in relation to the previous baseline design. Most of the design aspects that were preserved from the baseline device design are not discussed here at length.

As given constraints for the design work, the device hardware form factor was already finalized at the start of the text input method design process. The device would feature a touch screen, which was therefore to be used to provide all the text input methods to the user. The operating system software was based on the use of Debian Linux (v2.6) and GTK+, from which the basic UI components were imported.

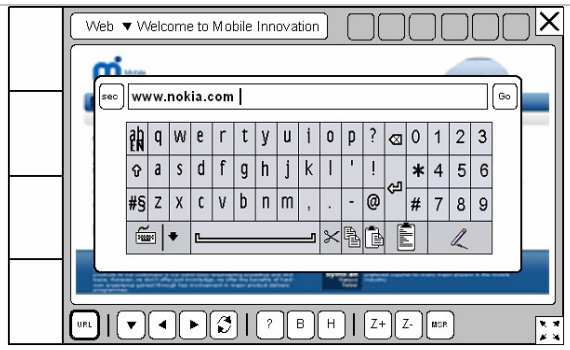
As with all software projects, additional restrictions came in the form of timetable and resource limitations. The design was to be specifiable and implementable within the given timeframe for the whole product, and with a reasonable use of available implementation personnel and other resources available for the project.

4.3.1 Design iterations

The final layout and design for the virtual keyboard input method is the end result of many rounds of design iterations and reviews. As a hopefully interesting artefact, here is a short collection of layout iterations from various stages of the user interface design process (see Table 1).

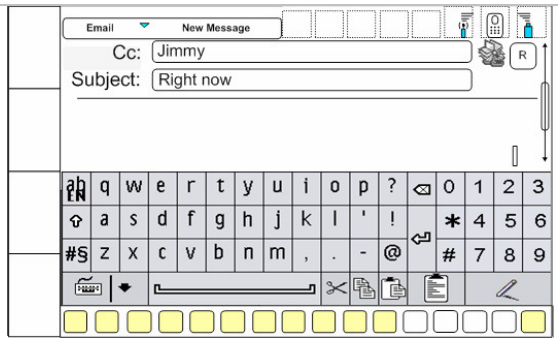
Table 1 Input method layout design iterations

One path of the design work started by trying to improve the baseline design by iteration. Here is a sample of an iterated virtual keyboard floating input method design.

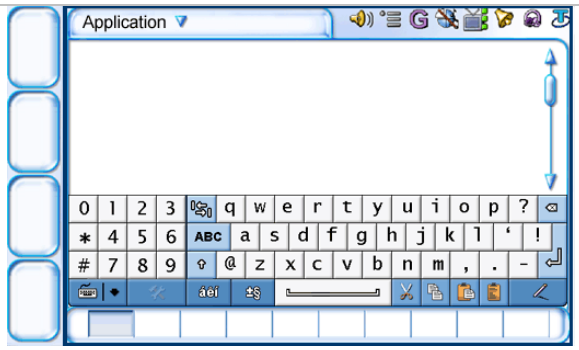


Another proposal was to embed the input method contents directly inside the dialogs where text input would occur, essentially embedding the input method always inside the current context or view.

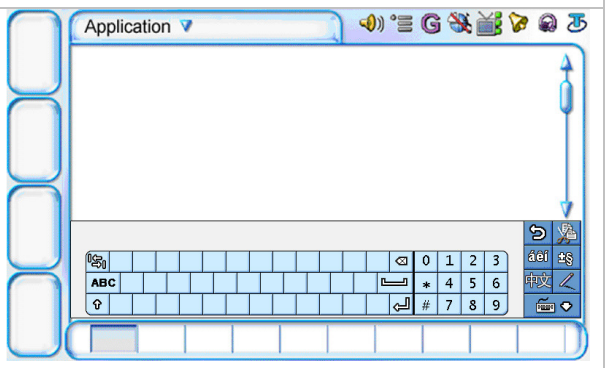
In one design study, to support views with multiple text areas, the input method would have contained arrow controls to switch directly to the next available text field.



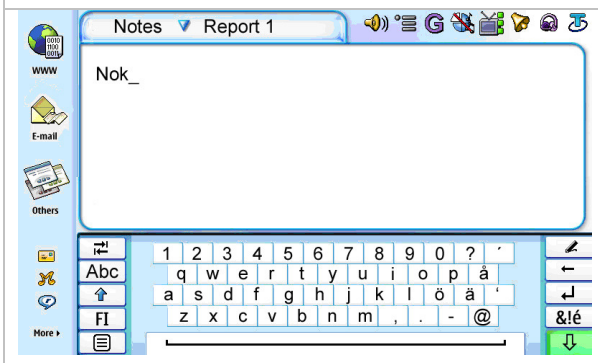
Ideas about the input method resizing the content area were also produced. Here the input method is still displayed above the application area toolbar in the bottom of the view.



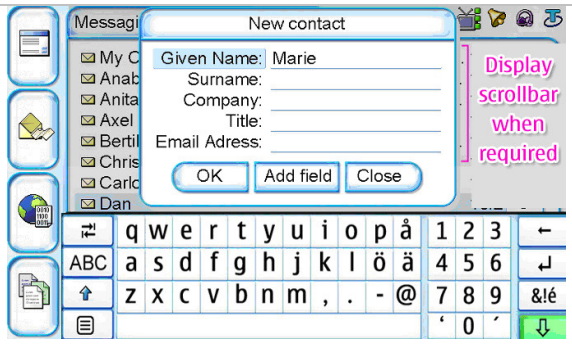
This concept featured different visualization for non-input buttons. This buttons would later be moved to the sides of the input method layout.



An iteration of the previous concept collected the common controls to the right side of the input method, leaving the basic form of the keyboard simple.



The design started to settle down on the fixed input method proposal. Here is a preliminary draft, with five vertical rows of controls and keyboard buttons. All application content, including the toolbar, is located above the input method area.



The final design is starting to take shape, with efforts here taken to minimize the vertical height and utilize the horizontal shape of the fixed input method area (reducing one vertical row from the input method).

The final iteration settles on the design option of having the input method in a fixed position on screen, resizing the application area accordingly. Details of the design are further described in the following chapters.

4.3.2 Input method launching and closing

For input method launching the design and method utilized in the Nokia 7710 seemed worth preserving: tapping to launch input method is a simple and effective system. Usability tests showed this to be initially hard to find for some users, but once discovered it is almost instantly learnable. The positive aspects of the design far outweigh the possible first-time user confusion. No adequate justifications to switch to alternative designs, such as a having a dedicated icon or control to launch the input method, were found.

Another topic relating to input method launching was whether the input method should be automatically launched in some instances or not. After lengthy consideration, the decision was made not to launch the input method automatically, but to make launching only occur when the user triggers it (by tapping a text input field). Again the key issue came down to learnability and predictability of the input method. It is impossible to predict accurately whether users wish to input text or not in all of the use scenarios. This prediction can be done reasonably well in certain scenarios, but not at all with others. The users would basically be annoyed if the device would launch the input method in a situation where the user does not wish to input text.

On the other hand, in the counter cases where the user would want to input text but the device assumes that he does not, the lack of launching could feel like a software bug, since the feature would work in some other places in the device. Different users exhibit different motivations and behaviours when using the applications of the device. As an end result, sometimes the input method would launch, sometimes it would not, without any apparently consistent logic. For instance, when opening a text document, does the user intend to read it or write to it? If the system elsewhere would provide automatic launching, designing the system in this instance either to launch or not to launch would make it seem to fail to predict the intentions for a part of the users. Some views also contain multiple text input fields. If the input method has a design with manual launching, then the selection of to which particular input field the user intends to input text is made explicit.

Additionally, the input method covers a large area of the total application area. If the input method is not automatically launched, the user is presented with a larger and clearer display

of the available content and options. Also use cases containing sub-dialogs prove problematic: if the input method would be launched when first entering a dialog, should it be again launched after a sub-dialog has been accessed and closed (especially considering that not all sub-dialogs might then require text input).

To summarize the issue, the conclusion was that a smaller penalty in terms of the total text input task would be to require the user to request explicitly to launch the input method, than the penalties and inconsistencies that automatic input method launching would cause. Manual launching gives a consistent user experience, where the user first views the current content and then explicitly selects to edit content in it.

No major changes were done to the rules regarding the closing of the input method. The input method closing should occur automatically once the user navigates away of the current context of inputting text. If the user switches from one input field to another, the input method window should not be closed in the meanwhile. Even though the input method window closes automatically, additionally a manual control for closing the input is also provided in the UI, both to give the users a sense of security and to give a consistent method for hiding the input method, to see more of the application view. No cases where the input method would be displayed without an input field where the inputted text would go to were to be permitted.

4.3.3 Input method layout

The design work started with the assumption of retaining the floating input method. Refinements were made to the baseline floating input method layout and its behaviour (see Figure 19).

Figure 19 – Nokia 770 – first design drafts, improved floating input method

Some of the usability issues could ostensibly be fixed by iterating the layouts and behaviour, but many critical problems persisted. After extensive studies and discussions, the decision was made to rethink the previous design completely, and eventually move from a floating window input method design to a fixed window area design.

In a fixed area design, the input method area does not overlap over the application content area. Whenever input is to be performed, the content area is rescaled smaller to make space for the input method (see Figure 20).

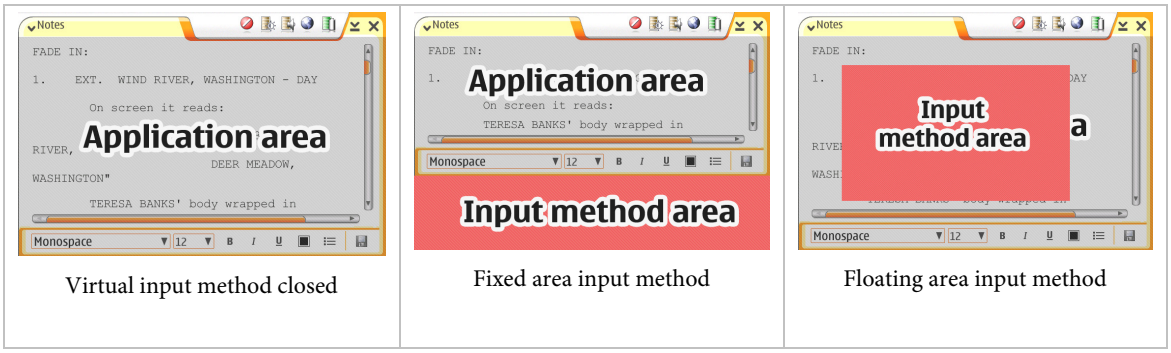


Figure 20 – Application area scaling in fixed vs. floating input window designs

A fixed input area design has many benefits compared to a floating input method. Having the input area fixed - always on the bottom of the view - and resizing the application area eliminates most of the problems that a floating input method area introduces. The input method element simply cannot overlap the content area anymore. This also eliminates the need of the user to manually reposition the floating input method and the content area just to be able to perform the input task (see Figure 21).

Figure 21 – Application area resizing with the fixed area window design

Since there is no overlapping, there is also no need to retain the pre-edit field in the input method design. All text input goes and appears directly in the location where the user intends to input it. This simplifies the design and implementation, and eliminates some potential elements of user confusion. A fixed input method always appears in the same position, making it consistent and easy to learn for the user.

When the content area is rescaled, all the content in the remaining area will always be completely displayed to the user. When there is no risk of window overlapping, application design needs to only make sure that whatever is displayed in the remaining area is relevant to the current text input activity.

As an additional benefit for the fixed area design, presenting the input method on the bottom of the view minimizes the hand obstructions when inputting text, since with this layout the hand of the user will not block the actual content area (see Figure 22).

Figure 22 - Hand obstruction of input method, with floating and fixed layouts

Naturally there are also drawbacks to a fixed layout input method. A floating input method window would make the job of the application user interface designer easier. With a floating input method design application views that contain text input elements do not have to be able to resize themselves elegantly, allowing for more complex, custom layouts with fixed component positions. Views that would not support scaling would also make them likewise simpler to implement than dynamically resizing views.

Another advantage of a floating input method window is the increased freedom in determining the shape of the actual input method area. Whereas with a fixed input method window the shape selection only tries to minimize the loss of available application area space

(resulting in a highly rectangular shape), a floating input window can take on a more square shape. When referring to Fitts' law, a square shape would be more advantageous in terms of reducing the amount of hand movement required. This comes noteworthy especially when reaching the threshold when the entire width of the keyboard cannot be reached simply by moving the fingers, but rather a more substantial movement of the entire hand is required. Also, with a floating input method the user would have the freedom in positioning the input method in a screen position that would feel ergonomically optimal. With a fixed position for the input area this determination is done on behalf of the user, whether it is the optimal position or not.

A fixed input area does result in significantly reducing the available application area. Fortunately this does not in most cases cause a major problem: for the purposes of text input it is not really necessary to have more than a couple of rows of the current input area visible at any time. A majority of text input tasks that the user performs with this device consist of the user only inputting one or two words at a time, writing long sentences and paragraphs with a touch screen input method is a rare occurrence.

Fixed input method requires also more support from the UI application framework, so that the application area can resize itself to make space for the input method. Likewise, the application UI designers would need to apply the scalability requirement to their designs.

It is relatively safe to say that introducing a fixed input method design needs to be done at the early stages of UI platform and style development. Migrating later on to a fixed design option, for instance from the Nokia 7710 floating input method design, would turn out to be very complex and troublesome, since it would require changes to all of the applications and layouts that have been designed and implemented so far. Especially for a well established platform, containing a large number of 3rd party applications, this work would be almost unfeasible.

In this project, the decision to go with a fixed input area design would have been even easier if the device display would have been in portrait format instead of landscape format. With a landscape format screen launching the input method creates a horizontally rectangular input and application area. Fortunately western languages are also written horizontally, from left to right, so the lack of vertical height isn't a critical issue as long as the horizontal application

area is preserved: even a few rows of text can comfortably fit and display the majority of text that the users will input with the device.

4.3.4 Text input method layouts

The next task was to design the actual control layouts of the virtual keyboard input method.

Since the device display form factor was landscape, and a fixed input method design occupies a horizontal segment of that area, the key target was to minimize the height of the input area, and to utilize the available complete horizontal area effectively.

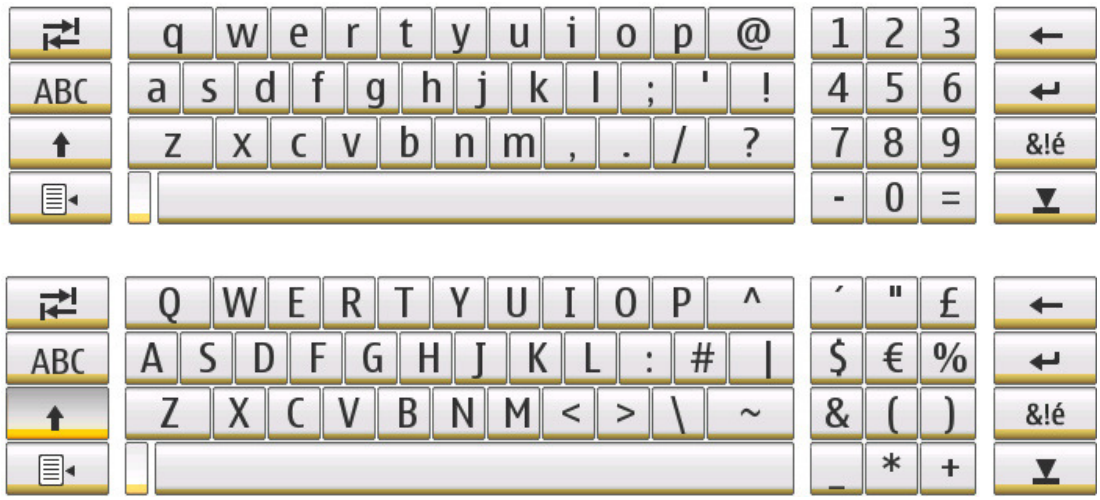


Figure 23 - Text input method, virtual keyboard upper and lower case layouts

The design concluded with having four rows of controls: four was the minimum number with which the input method design could still resemble a traditional physical keyboard while providing an adequate amount of keys for the user. The amount of character keys was iterated until a basic layout was found where a sufficient amount of keys – and no more – would be available for all the different languages that the input method was to be able to support.

The keyboard layouts in the device are based on the Qwerty layout. The button layouts are presented in a staggered format, to better resemble a physical keyboard. The virtual keyboard features a different keyboard layout for each of the supported input languages, for a total of 17 different keyboard layouts: English, French, German, Spanish, Portuguese, Dutch, Danish,

Norwegian, Swedish, Finnish, Greek and Russian, with some of the languages having multiple layouts for differing regional variants.

Since the number of keys on the virtual keyboard is less than with a full sized physical Qwerty keyboard, only the most important characters and symbols for each particular language are included in the layouts. The less frequently used characters and symbols are available in the special character views, accessible through a control on the right hand side of the layout (see Figure 24).

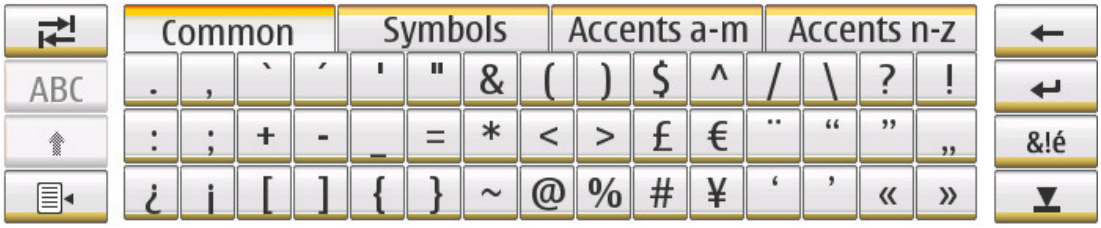


Figure 24 - Text input method, special characters view layout

As discussed in Chapter 2.5.1, the key size of one character button (approximately 4 millimetres in both width and height) for this implementation falls comfortably inside an acceptable zone for stylus usable virtual keyboards. Whereas the baseline design featured a separate number view, causing some problems of discovery and usage, this aspect could be improved by including the number keypad directly in the lower case key layout. The upper case layout contains commonly used special characters in the place of the numbers, similar to how they are accessible from a hardware keyboard by pressing Shift and the respective number key at the same time.

The two horizontal sides of the input method area contain commonly used controls, in a layout that tries to utilize knowledge of a conventional keyboard layout. The left side contains the tabulator, caps lock, shift and the input method menu keys. The right side contains the backspace, enter, special characters view and input method closing keys (see Figure 25). A separate input method popup menu was created to contain the less frequently used controls (for language selection, input method settings, cut/copy/paste and similar functions).

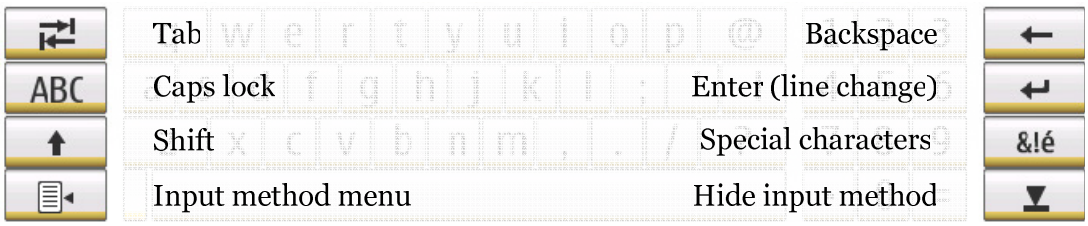


Figure 25 - Text input method, common control areas

The placement of the common controls also utilizes Fitts' law (and the mile-high menus) to a certain extent: the controls on the right hand side are easy to access, since they are at the edge of the display area. Space bar, which is the most frequently inputted character, is also easy to tap because it is located on the bottom of the screen.

As an additional note, the common control design is mostly similar between the virtual keyboard and handwriting recognition input methods, allowing the user to learn and utilize the same design with both of the input methods.

4.3.5 Text input and editing

The pre-edit area that was included in the baseline device UI design has been removed from this version. By changing the overall design from a floating window design to a fixed area design, the problems caused by window overlapping were removed. All text output occurs now directly to the text field that contains the cursor, similar to what the behaviour would be with a hardware keyboard. Although removing the pre-edit field has had generally a positive effect on simplifying the overall interactions, there is a small negative setback with the fact that the users are required to move their gaze a slightly longer distance between the keyboard area and the outputted text for input verification. In practice this hasn't proven to be a major problem, since the movement distance is still only a few centimetres.

Text editing follows common interface standards from desktop environments. The user is able to highlight a block of text from text input area and edit it. Common shortcuts for cut, copy and paste functions are also available, as well as the possibility to reposition highlighted text by directly dragging the highlighted block to another position on screen.

4.3.6 Assistive methods

The assistive method of sentence auto-capitalization design was preserved from the baseline device UI design, with slight adjustments to its behaviour. The keyboard layout is switched to upper case at the start of new sentences, unless the application explicitly specifies that the input should be in lower case. Mode detection is also provided, for instance in the numbers mode the layout with the numbers is displayed, and all the unsupported character keys are dimmed (to help the user to understand the constraints of the required input).

As a new assistive method, word completion was introduced for the device. The word completion feature predicts the word the user is attempting to input from the first inputted characters of that word. (See section 2.10 for discussion about word completion methods.)

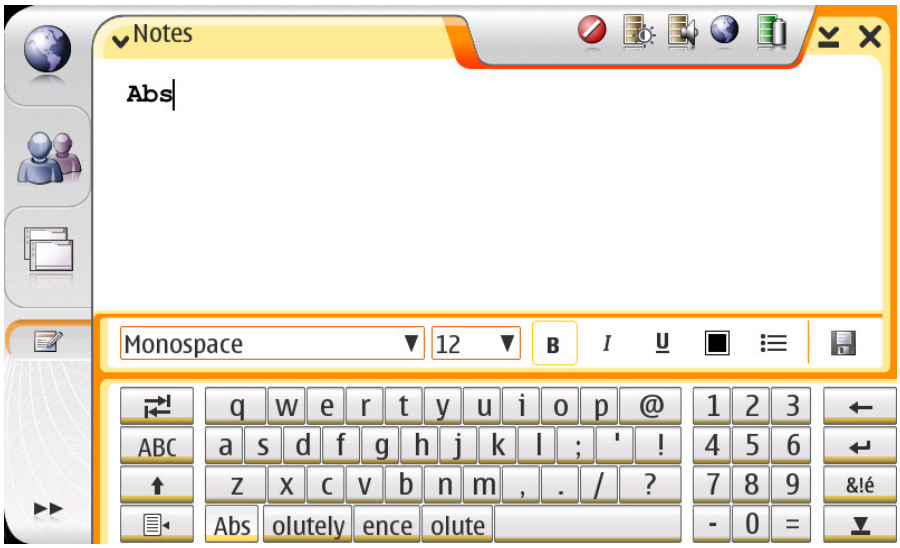


Figure 26 - Text input method, word completion presentation

Many interface designs found in other devices feature a popup window, or some additional presentation area, to display the contents of the word completion function. However, any such design would either requires the total input method area to be larger, or then the word completion candidates to fall outside the fixed input area, causing problems with overlapping which the fixed layout design exactly tries to solve.

After evaluating the different options, the word completion candidates were placed on top of the space bar. To utilize the available space in a more effective manner, the candidates are displayed so that the start of the word on which all the candidates are based on is displayed

once, and then the endings (suffixes) of the word are displayed, not the complete words (see Figure 26). By placing the candidates right next to the keyboard area, the aim was to make them easy to spot for the user.

This position for the candidates results in the input stylus flow where the user always finishes a word by tapping at the bottom of the view: either on one of the displayed word completion candidates, or then on the space bar area. Tapping a candidate inserts the candidate and a space, so that the user can directly continue with writing the next word. On a negative side, this design solution makes the space bar a smaller and partly a moving target.

The word completion feature was designed to feel and behave like a non-obtrusive feature. It offers candidates and functionalities for the user, but it does not demand the user to utilize them or to interact with the feature. Accordingly the behavioural rules for the word completion behaviour were specified to be simple and automatic. All words inputted by the user that are not yet contained in the system are added to the word completion dictionary, if they pass certain filters that test whether they are valid words (for instance not to contain special characters inside the word).

On a conceptual level, training the word completion feature is not a rewarding task to the average user, so demanding the user to manually train really wouldn't work with any prolonged use. Each word completion candidate that is selected by the user is reordered in the word completion library so that the next time the same phrase is entered the previously selected candidate appears as the first candidate.

Automatic addition of inputted words does result in the user occasionally adding typos to the dictionary, but this was considered to be a minor issue. The user interface does provide the user the opportunity to remove a word from the dictionary, but the automatic candidate reordering also eventually filters the incorrect candidates away from being displayed as the first candidates for the user.

Simple stylus gestures were also added on the virtual keyboard, for backspace, space, accessing the upper case variant of a character and line change. For instance backspace can be accessed by drawing a gesture to the left from on top of any character on the keyboard. Gestures are the best example of the interface design utilizing Fitts' law: the easiest target to select is the one that is already over the pointer.

4.4 Other changes

Special consideration was also placed in the design in reducing the required amount of text input. Reduction was done for example by suggesting sensible default values for text fields (for instance automatically forming a possible file name for a created document from the document contents) and by generally avoiding the use of the text input method when there were other alternative methods of performing actions. For instance a task like entering a certain date can be picked from a pop-up window displaying an interactive calendar element, instead of demanding the user to type the date in numbers.

Measures like these also assist in reducing the amount of errors: for instance by providing a calendar window popup errors with typing months and days incorrectly cannot be created anymore, since the popup can be designed so that it only displays and provides valid selections.

4.5 Implementation restrictions

The word completion feature was acquired as a licensed software component, and therefore there were certain limitations in controlling its exact behaviour. A reasonable compromise had to be made between what would have been the preferred behaviour, and what the licensed engine was capable of providing for the system. Also, the available software resources did not allow to design, acquire or to implement more comprehensive assistive methods, such as phrase prediction, error correction or spell checking.

Some of the original concepts featured the use of subtle animations, transitions and visual effects for the various on-screen elements, for instance to revealing and hiding the fixed input method from the bottom of the screen. These were also left unimplemented, due to scheduling and resourcing issues.

The input method is currently not aware to which application or task it is inputting text to, it only knows which type of input is expected (text, numbers etc.). This means that for instance instant content validation is not possible. With proper content validation, for example when

manually inputting a numerical value for a day, values under 1 and over 31 could be instantly spotted, corrected and the user notified. Being context-aware would allow many other more advanced possibilities: the contents of the virtual input method layout could adapt to better fit the input task in question, by displaying characters and character sequences that are often required in the task.

4.6 Overview of usability study results

The design described in the study has been involved in several usability tests, both specific to the input method and a part of usability tests involving other features in the product. The usability tests have been independently performed by external subcontractors. The full usability study documents are not public material, but a summary of the main findings of the various usability studies can be presented.

In general, no critical usability problems have been found with the virtual keyboard design. When measuring overall satisfaction with individual features in the device, the virtual keyboard gets above average scores in this scale. Novice users averaged an input speed of 1.2 characters per second with the virtual keyboard; this translates to an input speed of about 10 words per minute. (As a non-scientific sidenote, I as the designer and as an experienced user can average, without the use of word completion, input speeds of approximately 15 words per minute with the virtual keyboard input method of the 770 Internet Tablet.)

Going through the list of study results, the following issues have been found as the biggest problems with the current design: Some of the users have problems in first time use in finding how the virtual input method is launched, since there is no indication of how to launch it on screen. Part of this problem has been caused by having a blinking cursor displayed when opening certain views and dialogs. In this case the users might expect to be able to input text to the position of the cursor, and they did not think that they would have to tap the cursor that is already visible.

Some users had trouble finding the numbers on the virtual keyboard, since they are only available in the lower case layout. Some users had problems accidentally selecting the candidates displayed in the word completion area when they intended to press the space bar:

this is understandable, since the total width of the space bar varies constantly when inputting new text. This was the compromise that was made when selecting to display the word completion candidates inside the input method area.

Some users had difficulties with understanding the design with the Shift and Caps lock on the device: they pressed the ABC button when they desired to utilize the function the Shift button would provide. "ABC" is a well known indication of text case from other Nokia devices, and this probably attracts users to assume that it should be used for the purposes that the Shift key would provide.

When discussing about the overall use of the device, the virtual input methods were criticized for not being usable when walking with the device. Another criticism was about the fact that they require two hands and the use of the stylus when inputting text. Some of the mobile phones the users were using did permit one-handed text input as well as inputting text while walking.

Considering the overall test results, most of the found issues are first-time user issues, learnable after the input method for a short period. Some of the found issues were already addressed in the 2006 software version of the platform. Unfortunately no usability test has been currently performed where an exactly similar input task would be performed with the Nokia 7710 and the Nokia 770 Internet Tablets, so no direct comparisons are possible with the baseline device.

5. CONCLUSIONS AND DISCUSSION

5.1 Work results

The design work described in the thesis was released in the Nokia 770 Internet Tablet product, in its Internet Tablet 2005 software edition. Along with the overall input framework and the virtual keyboard design described in the thesis, the Internet Tablet 2005 software edition contains handwriting recognition as an alternative touch screen input mechanism.

The text input method implementation was done on the basis of a written user interface specification. Individual localized keyboard layouts were also created to suit each language or locale that the device officially supports. Along the actual UI and layout specifications, this thesis document has been written, containing a summary of the most relevant findings of the study, design and research performed while working on this project.

5.2 Research conclusions

One of the key experiences out of this project is the realization that designing the text input method of a new user interface platform needs to be done at the very start of the overall design work. Especially if the platform plans to be able to provide text input also some means of on-screen or touch screen interactions, the ability to be able to integrate this text input method into the overall user interface structure of the platform and the individual applications at the very start is extremely beneficial.

In the case of the Nokia 770 Internet Tablet, only through solving and addressing these design issues and software requirements at the very start of the project, was it possible to provide a consistent and integrated solution to touch screen text input. Adding the touch screen text input method afterwards to an existing platform is a severe problem, especially if the platform and UI design has not been initially designed to be scalable to different screen sizes and display aspect ratios.

Another conclusion from the research work relates to the use of predictive methods for text input. Although many advanced assistive and predictive methods have been created and

proposed for various styles of text input, these can easily run into problems with requiring too much attention and interaction from the user, turning the benefits they might provide into drawbacks and annoyances. Including even a simple step of evaluation changes the nature of the input method quite dramatically. To exemplify this issue in a slightly aggravated manner: One analogy that could be offered to the task of text input is the task (and art) of playing the piano. The piano is a non-predictive instrument, an input method for music, of sorts: the player needs to press all the keys of the piano in exactly the right order.

Music, much like language, is not random, and predictive methods could in theory also be designed for the piano: auto-completion candidates for musical phrases, and error corrections for "obviously" incorrect notes. Naturally the language of music does not hold so strict grammar rules as our natural languages, but this analogy still serves a certain point. A proficient piano player, just like a proficient touch typist, thinks and executes phrases much faster than what would be possible by constantly observing and interacting with the input system, be it the piano display or the mobile device display.

And even if the interaction would not be slower than what direct non-predictive input would be, a piano player would find it hard to concentrate on *what* he is playing, much like a person expressing themselves with writing would find it hard to think about what he is saying, if he would have to constantly interact with the input method, think about what it is doing, observe incorrect candidates and pick the correct ones. An input method, be it the piano or the keyboard, can eventually transform itself to be a tool, a natural extension and expression mechanism of the user, but this is only possible if it can be eventually utilized without requiring constant attention of how to actually operate it.

The piano analogy is probably a bit extreme: predictive features like auto-capitalization and word completion candidates are useful in some cases, especially if they help to reduce the amount of required key or stylus presses. Many of the currently popular methods, such as T9, help to bring a satisfactory text input speed to devices that would otherwise feel far too slow to utilize. And as predictive methods increase their understanding of language and grammar, their usefulness could increase. Still the basic point may be valid: the less cognitive effort and interaction a input method requires, the more natural utilizing it will eventually become. Predictive methods may have a finite limit on the text input speed that it achievable, caused

by the additional time that verification takes, and this upper value is probably smaller than what is achievable by a non-predictive miniature Qwerty keyboard.

As described by this thesis, there still currently are no mass market mobile text input solutions that would offer performance or comfort levels similar to what desktop environments can offer to the user. However, this does not need to be seen as an insurmountable problem to designing mobile devices. It is next to impossible for mobile devices to compete against desktop devices in metrics such as information throughput, since the both the input and output capabilities are severely limited, both through technological and device physical properties.

Rather than focusing on tasks where desktop devices have clearly better properties, mobile device design should focus on the areas and use cases which desktop devices are unable to provide: linking and augmenting networked information and communication capabilities with the current physical reality of the user. Considering the task of text input, it is reasonably safe to predict that text input on a small portable mobile device will never reach the same input speeds or comfort levels as input with a dedicated desktop device or portable computer. Therefore the aim of mobile text input should not be to try to replace the fully-fledged word processing tasks desktop computers currently provide, but rather to provide sufficient text input capabilities in environments where full desktop solutions are not available, utilizing the already available skills of the user whenever this is possible.

Fundamentally, the choice between hardware and software (virtual) input methods can be seen to be linked with the intended use of the device. If the device is intended to give the users the ability to perform tasks that rely heavily on text input, a hardware input solution is probably warranted, simply in order to be competitive with other devices that are utilizing hardware keys. If the mobile device has no touch screen, but still requires text input from the user, it is also obvious that some sort of solution utilizing hardware buttons is needed.

On the other hand, if the key intended use cases of the device do not feature heavy use of text input, then also the demand for a comparatively effective input method for this device is decreased. Any solution consisting of hardware buttons is a relatively expensive and complicated piece of mechanical hardware, adding which would then increase the cost and also the physical size of the device. Any portion of space that the keys take is away from the

space that would be available for the screen surface area. Especially if the device in question already has a touch screen, it becomes much more justified to attempt to provide text input only through the touch screen.

5.3 Discussion on the scope of the study

When considering the scope and limitations of the thesis (as described in chapter 1.3), the scope of the research was highly targeted towards the actual practical design target in the project, a mobile touch screen device. Even with the limited scope on some issues the written research needed to be limited to focusing only on the key aspects. The research data did turn out to be highly relevant to the actual work at hand.

Some parts of the research do have limited applicability also outside mobile touch screen devices. For instance certain assistive input methods are already available on desktop environments, probably the most noteworthy of these being the spell checking and auto-correction features that some word processing applications feature. Their current level of design and implementation inside these applications on a reasonable level, but they are not utilized consistently throughout all of the applications, i.e. it is not an operating system level feature. However, as already discussed, there is only a limited need for these kinds of features, since non-predictive input can be very fast with full sized physical keyboards. Probably nearly every user has also had some negative experiences with word processing applications also automatically "correcting" things that do not need corrections.

Scoping the thesis to concentrate on Western languages was a relatively easy decision, but it did simplify some of the overall conclusions a considerable amount. Especially the issues relating to predictive vs. non-predictive input become far more complicated when starting to consider input languages such as Japanese or Chinese. Many of the rules and findings in this thesis cannot be taken for granted when moving outside the scope of Western languages.

Even though currently touch screen text input methods have little relevance in the desktop environment, this might change in the future. Currently touch screens are available only on certain laptop computers, but there are signs that this technology will be introduced also into a number of conventional – as well as some completely new – desktop computing form factors. In this kind of use, when using the touch screen, solutions for short bursts of text input (only a couple of characters) might be needed also on the touch screens, even if these

computers would also carry a full sized physical keyboard. For these kinds of use cases even rather radical designs with predictive or recognition methods might be warranted.

5.4 Further improvement areas of the design

Looking into the future, many areas for further research and design remain. If considering the current design and implementation, the issues found in the usability studies (as discussed in chapter 4.6) should be discussed and attempted to be corrected, if possible.

Looking at the specifics of the current design, the special characters view could be improved in many ways, for instance by allowing user customization of the view, or providing the most commonly used special characters in a more effective manner. The word completion feature display format (see chapter 4.3.6, displaying truncated candidates on top of the space bar), while providing efficient usage of available screen space, is also cognitively more challenging, and its positioning interferes with pressing the space bar.

The behaviour of the word completion engine could be enhanced in several ways. The current prediction rules and candidate presentation logic is effective, but needlessly limited. A smarter word completion feature would predict candidates based on entire sentences that the user has written, as well as it would store the frequencies of usage for each word completion candidate, which would allow to improve the engine behaviour over time. Next to the current word completion feature, other assistive and predictive features, such as next word prediction, spell checking, manual or automatic error detection and correction could also be introduced to the system. In addition to this, the contents of the basic language dictionaries could be updated to contain words and names frequently used in an Internet communication device.

As discussed in chapter 2.13, a virtual input method gives far better possibilities to adapt the method for each available input task than a hardware key solution. The current implementation is context-aware only to a very limited extent: it carries no knowledge of which application or task it is requested for, only knowledge is the requested type of input (normal text, numbers, passwords, web addresses etc.). For instance, when inputting a web address to the web browser of the device, the presented input method could adapt itself so

that it would provide functions and text shortcuts relevant for that input task. Context-awareness has potential, but it also creates risks in the form of decreased stability and increased need of learning from the user.

Instant error validation (as discussed in chapter 4.5) could also be performed by better context awareness of the input method: if the input method would know what to expect, it would be better able to note the user of errors and possibly automatically correct the inputted text to the correct form.

The current touch screen input methods have been designed to be operated with the stylus that ships with the device. However, users also attempt to use the device with their fingers, without taking the stylus into use. Providing a finger usable keyboard would allow the users to input small amounts of text without having to take the stylus into use. (A separate finger input keyboard has been designed for the next software version of this product, the Internet Tablet 2006 software edition.)

5.5 Further research and development areas

One of the major drawbacks of a touch screen text input method is the lack of tactile feedback, both when selecting the controls and when pressing them. This feedback could be provided with a separate mechanism in the device: for instance Poupyrev et al. discuss directions and strategies for utilizing haptic feedback in pen computing, utilizing piezoelectronics as the technology. Their preliminary results indicate clear gains with the use of technology, especially with harder tasks (small on-screen controls) and with the subjective preferences of the users. (Poupyrev et al., 2004) These technologies have great potential in reducing the negative issues that the lack of haptic feedback currently creates to any touch screen interactions. Next to the use of piezoelectronics, the use of vibrational motors holds some of the similar potential. Another research area would be the use of auditive feedback to indicate both successful and unsuccessful text input actions.

The current touch screen implementation of the device can function only with one touch point at the same time. Research and development into multi-touch screens, i.e. screen that recognize two or more separate touch points at the same time is yielding some very interesting results. Also the first commercial products with multi-touch capabilities are starting to come to the market. The potential of multiple simultaneous or parallel inputs, in

any form (also through multimodal input), to improve the usability and efficiency of a mobile device can be extensive. Naturally the small size of a mobile device imposes some limits of the potential of this technology: if the touch screen only fits two fingers at the same time, a proper multitouch solution might not prove very beneficial.

Another interesting research field relates to the amount and nature of layouts utilized in the input method: if an input method provides multiple layouts, especially if such layouts are dynamical in their nature, then the learning requirements are drastically increased compared to traditional physical keyboard. For example, suppose a virtual input method would provide a specific layout for the scenario where the user is entering an email address. Only those characters and symbols that would be valid for an email address would be provided and displayed. Such a layout would in theory be a very effective design, since no layout switching would be needed anymore. Then again, it would be more or less different from the other layouts that the virtual input method would be providing. With such a design issues with skill transference and conflicts with previously known layouts and patterns could easily start to arise. An interesting research field would be linked to minimizing these conflicts and discovering the best possible compromise between dynamical input method layouts versus the negative effects that having multiple different layouts would cause.

Another issue relating to the layouts has to do with the various optimized keyboard layouts. A *de facto* standard for an optimized keyboard layout hasn't yet arisen. The current designs have mostly concerned themselves with optimizing their theoretical performance in the form of input speed. Further usability research in such methods and layouts could be warranted. The goal would be to come up with a reasonable compromise between providing a substantially improved maximum input speed while still being effective and easy to learn and remember. Many of the current optimized designs do not even include a full range of characters (numbers, special characters, symbols) and functions (text editing operations, cut/copy/paste commands etc.) that a fully functional virtual input method would require. This is not to say that such methods do not exist at all, but rather the current candidates seldom come from the field of academic search, and the quality of their designs vary to a great extent.

5.6 Reflections on text input methods

As discussed in Chapter 2.11, the selected design and the Qwerty layout for the virtual keyboard is nowhere near the current fastest keyboard designs for a stylus keyboard. An optimized keyboard would allow an over 50% increase in theoretical maximum input speeds for the stylus compared to the selected Qwerty keyboard layout. Still, a 50% increase in speed (or even more) could also be obtained for instance by adding a miniature hardware Qwerty keyboard to the same given device, similar to what for instance the Nokia 9300 Communicator device does.

There currently really aren't any shortcuts or silver bullets around this problem: with the currently available technology, a device developer cannot provide a touch screen Qwerty layout based input method that would provide the same input speeds as a properly designed hardware key input method would provide, despite any marketing claims to the contrary. Advances in touch screen technology, especially in the areas of haptic feedback and new innovative materials, can in the future decrease the gap between hardware and software solutions, but there is ample reason to be sceptical about any possibility to reach the same level of speed, not to mention about claims of surpassing it.

When comparing Qwerty with an optimized layout, it is reasonably safe to state that the general user preference would be to utilize a familiar layout (such as Qwerty) than to take the time to learn a completely new input method layout and functionality. This would be especially true when there would be none or limited skill transfer possibilities to utilize the learned layout in other devices. The Nokia 770 device was not intended for tasks that would feature heavy text input. Since the device was designed for only occasional use, it would be unreasonable to expect the user to be willing to learn a non-standard text input method design just in order to use this particular device.

As previously mentioned, a *de facto* standard to an optimized keyboard layout doesn't unfortunately currently exist. If one would exist, then it would have been justifiable to study adding such a layout to the device, since one could assume that some users would already know this input method beforehand, and people taking the effort to learn this layout could then utilize their knowledge also in the future, with other similar devices.

Finally, I would like to give a personal viewpoint to the matter of keyboards and Qwerty layouts. I took touch typing courses in lower secondary school. At that time, after two years of training I achieved text input speeds of about 40-50 wpm. My current writing speed with a full Qwerty keyboard averages in the 70-80 wpm range. For me, touch typing is a very useful skill exactly because the Qwerty layout is the standard layout. In my lifetime I've used probably hundreds of different input devices, both desktop and mobile, with virtually all of them using fundamentally the same Qwerty layout.

Even if Dvorak would be theoretically a significantly more effective layout for a hardware keyboard, or a Metropolis optimized layout for a virtual keyboard, if I would take the time to learn to use those layouts, for instance on my home computer, learning these skills would impede my typing on other Qwerty layout devices that I would be using. The skill interference between the two input methods would be substantial.

The ecosystem of input devices is growing constantly. This, along with the rate of technological development leads to the fact that devices are quickly taken into use and later on quickly replaced with other similar devices. For this to be possible standards in general user interaction and interface design are extremely beneficial, and text input is a natural part of these standards.

Instead of learning a new input method for every device, previous knowledge is a valuable commodity which should not be left unutilized. The same way as learning to drive one car should give a person the ability to drive nearly all cars, learning to type on one keyboard should be a reusable skill. At the end of the day the benefits of having a common standard and layout far exceed the negative sides of such standards being theoretically less than optimal in some manners.

Balancing this conservative approach with the need and potential for further innovation makes the field of mobile text input makes an exciting field of study. Mobile devices are fulfilling the promise of personal computing on a completely new level, being nearly always available and always connected to the rest of the world. The experiences can be much more intimate and the benefits tangible, but likewise the demands and challenges for such interfaces become harder to master.

REFERENCES

Accot, J., Zhai, S. (2003)
Refining Fitts' law models for bivariate pointing
SIGCHI conference on Human factors in computing systems (CHI '03) (pp.193-200). FT. Lauderdale, Florida, United States: ACM Press. ISBN: 1-58113-630-7.

Carroll, J.M., Rosson, M. B. (1987)
The paradox of the active user
In J.M. Carroll (Ed.), Interacting Thought: Cognitive Aspects of Human-Computer Interaction (pp.80-111). Cambridge, Massachusetts: MIT Press. ISBN: 0-262-03125-6.

Clarkson, E., Clawson, J., Lyons, K., Starner, T. (2005)
An empirical study of typing rates on mini-QWERTY keyboards
Extended abstracts on Human factors in computing systems (CHI '05). Portland, Oregon, United States: ACM Press. ISBN: 1-59593-002-7.

Eatoni Ergonomics (2007)
Products - Eatoni
Online document: Eatoni.com, from Eatoni Ergonomics [accessed May 20 07]. Available at: <http://www.eatoni.com/wiki/index.php/Products>

Gingrande, A. (2003)
A brief history of data entry
Online document: FindArticles, updated Oct 2003 [accessed Sep 05 2006]. Available at: <http://www.findarticles.com/p/articles/mi_qa3947/is_200310/ai_n9337111>

Gong, J., Tarasewich, P. (2005)
Alphabetically constrained keypad designs for text entry on mobile devices
SIGCHI conference on Human factors in computing systems (CHI '05) (pp.211-220). Portland, Oregon, United States: ACM Press. ISBN: 1-58113-630-7.

Gong, J., Haggerty, B. & Tarasewich, P. (2005)
An enhanced multitap text entry method with predictive next-letter highlighting
SIGCHI conference on Human factors in computing systems (CHI '05): Extended abstracts (pp.1399-1402). Portland, Oregon, United States: ACM Press. ISBN: 1-59593-002-7.

Grossman, T., Balakrishnan, R. (2005)
A probabilistic approach to modeling two-dimensional pointing
ACM Transactions on Computer-Human Interaction (TOCHI), Vol. 12, No. 3, September 2005 (pp.435-459). University of Toronto, Canada: ACM Press. ISSN: 1073-0516.

Harris, J. (2006)
Giving You Fitts
Online document: Jensen Harris: An Office User Interface Blog, created Aug 22 2006 [accessed Sep 15 2006]. Available at: <http://blogs.msdn.com/jensenh/archive/2006/08/22/711808.aspx>

Himberg, J., Häkkilä, J., Kangas, P. & Mäntyjärvi, J. (2003)
On-line personalization of a touch screen based keyboard
Proceedings of the 8th international conference on Intelligent user interfaces (IUI '03) (pp.77-84). Miami, Florida, United States: ACM Press. ISBN: 1-58113-586-6.

Isokoski, P., MacKenzie, I. S. (2003)
Combined model for text entry rate development
SIGCHI conference on Human factors in computing systems (CHI '03): Extended abstracts (pp.752-753). Ft. Lauderdale, Florida, United States. ISBN: 1-58113-637-4.

Isokoski, P. (2004)
Manual Text Input: Experiments, Models, and System
Academic dissertation. Department of computer sciences, University of Tampere.
Electronic dissertation, available at: <http://acta.uta.fi/pdf/951-44-5959-8.pdf>

Jacob, R.J.K. (1995)
Eye Tracking in Advanced Interface Design
In Virtual Environments and Advanced Interface Design, ed. by W. Barfield and T.A. Furness (pp.258-288). Oxford University Press, New York.

James, C.L., Reischel, K. M. (2001)
Text input for mobile devices: comparing model prediction to actual performance
SIGCHI conference on Human factors in computing systems (CHI '2001). Seattle, Washington, United States: ACM Press. ISBN: 1-58113-327-8.

Jerz, M.
Technical specification – Nokia 7710
Online document: My-Symbian.com [accessed Sep 10 2006]. Available at:
<http://www.my-symbian.com/s90/techdata_7710.php>

Liebowitz, S. J., Margolis, S. E. (1990)
The Fable of the Keys
Online document: Journal of Law & Economics vol. XXXIII (April 1990) [accessed Sep 23 2006]. Available at:
<http://wswwpub.utdallas.edu/~liebowit/keys1.html>

Kober, H., Skepner, E., Jones, T. Gutowitz, H. & MacKenzie, I.S. (2001)
Linguistically optimized text entry on a mobile phone
Online document: unpublished, submitted to CHI 2001 conference. Eatoni Ergonomics Inc. [accessed May 17 2007]. Available at: <http://www.eatoni.com/research/chi.pdf>

Kristensson, P. (2005)
Breaking the laws of action in the user interface
SIGCHI conference on Human factors in computing systems (CHI '05): Extended abtracts (pp.1120-1121). Portland, Oregon, United States: ACM Press. ISBN: 1-59593-002-7.

Kristensson, P., Zhai, S. (2005)
Relaxing stylus typing precision by geometric pattern matching
Proceedings of the 10th international conference on Intelligent user interfaces (IUI '05) (pp.151-158). San Diego, California, United States: ACM Press. ISBN: 1-58113-894-6.

Költringer, T., Grechenig, T. (2004)
Comparing the immediate usability of graffiti 2 and virtual keyboard
SIGCHI conference on Human factors in computing systems (CHI '04): Extended abstracts (pp.1175-1178). Vienna, Austria: ACM Press. ISBN: 1-58113-703-6.

Magnien, L., Bouraoui, J. & Vella, F. (2003)
Using visual cues to help text entry on PDAs
Proceedings of the 15th French-speaking conference on human-computer interaction (pp.252-255). Caen, France: ACM Press. ISBN: 1-58113-803-2.

MacKenzie, I. S., Buxton, W. (1992)
Extending Fitts' law to two-dimensional tasks
SIGCHI conference on Human factors in computing systems (CHI '92) (pp.219-226). Monterey, California, United States: ACM Press. ISBN: 0-89791-513-5.

MacKenzie, I. S., Tanaka-Ishii, K. (2007)
Text entry systems
Morgan Kaufmann. ISBN: 0-12-373591-2.

MacKenzie, I. S., Buxton, W. (1993)
A tool for the rapid evaluation of input devices using Fitts' law models
ACM SIGCHI Bulletin (pp.58-63). Vol. 25, No. 3, 1993. New York, New York, United States: ACM Press. ISBN: 0736-6906.

MacKenzie, I. S., Zhang, S. (1999)
The design and evaluation of a high-performance soft keyboard
Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit (CHI '99) (pp.25-31). Pittsburgh, Pennsylvania, United States: ACM Press. ISBN: 0-201-48559-1.

MacKenzie, I. S., Kober, H., Smith, D., Jones, T. & Skepner, E. (2001a)
LetterWise: prefix-based disambiguation for mobile text input
Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01) (pp.111-120). Orlando, Florida, United States: ACM Press. ISBN: 1-58113-438.

MacKenzie, I.S., Zhang, S. X. (2001b)
An Empirical Investigation of the Novice Experience With Soft Keyboards
Behaviour & Information Technology. Vol. 20, No. 6 (pp.411-418).

McGuffin, M. J., Balakrishnan, R. (2005)
Fitts' law and expanding targets: Experimental studies and designs for user interfaces
ACM Transactions on Computer-Human Interaction (TOCHI), Vol. 12, No. 4, December 2005 (pp.388-422). University of Toronto, Canada: ACM Press. ISSN: 1073-0516.

Mizobuchi, S., Chignell, M. & Newton, D. (2005)
Mobile text entry: relationship between walking speed and text input task difficulty
Conference on Human computer interaction with mobile devices & services (MobileHCI '05) (pp.122-128). Salzburg, Austria: ACM Press. ISBN: 1-59593-089-2.

Nokia (2007)
Nokia 7710 Series 90 widescreen smartphone (image)
Online document: Nokia.com [accessed May 01 2007]. Available at:
<http://www.nokia.com/A4136017?category=7710>

Nokia (2007)
Nokia 770 Internet tablet (image)
Online document: Nokia.com [accessed May 01 2007]. Available at:
<http://www.nokia.com/A4136017?category=770>

Norman, D. A. (1988)
The Design of Everyday Things
New York, Doubleday/Current Ed. ISBN: 0-465-06709-3.

Noyes, J. (1998)
QWERTY – The immortal keyboard
Computing & Control Engineering Journal. Vol. 9, Issue 3, 1998 (pp.117-122). ISSN: 0956-3385.

Parhi, P., Karlson, A. K. & Bederson, B. B. (2006)
Target Size Study for One-Handed Thumb Use on Small Touchscreen Devices
Conference on Human computer interaction with mobile devices & services (MobileHCI '06) (pp203-210). Helsinki, Finland: ACM Press. ISBN: 1-59593-390-5.

Planar Systems, Inc. (2007)
Choosing the right touchscreen
Online document: Planar.com. Retrieved on 17.05.2007. Available at:
<http://www.planar.com/products/touchscreen/pt/choosing_touch_screen.html>

Poupyrev, I., Okabe, M., Maruyama, S. (2004)
Haptic feedback for pen computing: directions and strategies
Conference on Human factors in computing systems (CHI '04) (pp.1309-1312). Vienna, Austria: ACM Press. ISBN: 1-58113-703-6.

Ryu, H., Cruz, K. (2005)
LetterEase: Improving text entry on a handheld device via letter reassignment
Conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction (OZCHI '05) (pp.1-10). Canberra, Australia: Computer-Human Interaction Special Interest Group (CHISIG) of Australia. ISBN: 1-59593-222-4.

Sears, A., Revis, D., Swatski, J., Crittenden, R. & Schneiderman, B. (1993)
Investigating touchscreen typing: the effect of keyboard size on typing speed
Behaviour and Information Technology. Vol. 12, No. 1, 1993. Great Britain: Taylor and Francis. ISSN: 0144-929X.

Sears, A., Zha, Y. (2003)
Data Entry for Mobile Devices Using Soft Keyboards: Understanding the Effects of Keyboard Size and User Tasks
International Journal of Human-Computer Interaction. Vol. 16, No. 2 (pp.163-184).

Segan, S. (2007)
SureType
Online document: blackberry.com, from Research in Motion Limited [accessed Apr 22 2007]. Available at: <http://www.blackberry.com/products/suretype/index.shtml>

Shieber, S. M., Baker, E. (2003)
Abbreviated text input
Conference on Intelligent user interfaces (IUI '03) (pp.293-296). Miami, Florida, United States: ACM Press. ISBN: 1-58113-586-6.

Silfverberg, M., MacKenzie, I. S., Korhonen, P. (2000)
Predicting Text Entry Speed on Mobile Phones
SIGCHI conference on Human factors in computing systems (CHI '00) (pp.9-16). New York, United States: ACM Press. ISBN: 1-58113-216-6.

Smith, B. A., Zhai, S. (2001)
Optimised Virtual Keyboards with and without Alphabetical Ordering-A Novice User Study
INTERACT'2001 – IFIP TC13 International Conference on Human-Computer Interaction. Tokyo, Japan (pp.92-99). Available at: <http://www.almaden.ibm.com/u/zhai/papers/Softkeyboard/Alpha10.pdf>

Tegic Communications, Inc. (2006)
T9 Text Input – [Learn to Use T9]
Online document: T9.com, from Tegic Communications [accessed May 20 07]. Available at: <http://www.t9.com/learn/>

Wikimedia Commons
Keyboard layout images
Online document: Wikipedia.org [accessed Mar 20 2007]. Available at:
<http://en.wikipedia.org/wiki/Image:KB_United_States_Dvorak.svg>
<http://en.wikipedia.org/wiki/Image:KB_Sweden.svg>

Zhai, S., Hunter, M. & Smith, B. A. (2000)
The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design
ACM symposium on User interface software and technology (UIST '00) (pp.119-128). San Diego, California, United States: ACM Press. ISBN: 1-58113-212-3.

Zhai, S., Sue, A. & Accot, J. (2002)
Movement model, hits distribution and learning in virtual keyboarding
SIGCHI conference on Human factors in computing systems (CHI '02) (pp.17-24). Minneapolis, Minnesota, United States: ACM Press. ISBN: 1-58113-453-3.