Using Sketches to Edit Electronic Documents

Pedro Miguel Coelho Gonçalo França Faria
Instituto Superior Técnico
Av. Rovisco Pais, 1000 Lisboa
{pmc,gfs}@mega.ist.utl.pt

Daniel Gonçalves Joaquim A Jorge
Computer Science Department, IST
Av. Rovisco Pais, 1049-001 Lisboa
djvg@gia.ist.utl.pt, jorgej@acm.org

Abstract

Calligraphic interfaces mimic the interaction of users with pen and paper, to which most persons are used to from an early age. This makes them a privileged way of interacting with computers in efficient and natural ways. One of the tasks more often made with pen and paper is to write and correct text documents. This is done resorting to several symbols with specific meanings, representing the changes to be made in the document. It should be possible to correct electronic texts in a similar way, using calligraphic interfaces. In this paper, we describe the CaliEdit text editor, currently being developed for PalmOS devices with that goal in mind. In its development, we followed a user-centered design approach. The task analysis phase, with the help of questionnaires, identified the most common symbols for the most common text-correcting tasks. The editor itself uses the CALI shape and gesture recognizer, ported to PalmOS, to help recognize those symbols. Several porting and implementation problems were overcome. Currently, heuristic and usability evaluation studies are underway to help validate the editor.

Keywords

Calligraphic interfaces, gesture recognition, mobile devices, text editing, user-centred design.

1. INTRODUCTION

Calligraphic interfaces are a novel way of interacting with computing devices. They try to mimic the ways in which users interact with paper documents, allowing them to enter and manipulate data simply by drawing or sketching. Several heuristics and constraints can then be used to correctly interpret the users' wishes. This allows a more efficient, streamlined and natural interaction. Indeed, most persons are used to interacting with pen and paper from an early age. Also, paper allows a more flexible interaction than usual computer peripherals, such as keyboards and mice. Furthermore, even computer-unskilled persons can use pen and paper efficiently. This makes calligraphic interfaces suitable for a wide range of users and situations where the limited traditional interaction ways could pose serious problems.

Calligraphic interfaces have been used successfully in several applications in the past. The Java SketchIt project [CAETANO'02] allows dialogue boxes to be created from rough sketches of their composing elements. GIDeS [PEREIRA'03] provides a mechanism for the creation of technical drawings from sketches, with the help of spatial constraints. They have also been used for the retrieval of technical drawings, from rough sketches of what is to be found [FONSECA'03].

Until recently, special hardware such as touch screens or digitizing tablets was required to use calligraphic interfaces. However, with the dissemination of Personal Digital Assistants (PDAs) and the appearance of Tablet PCs, that hardware is now becoming commonplace. Those devices have interfaces based on direct manipulation of elements on the screen using a special pen or stylus (although often mimicking a mouse). The infrastructure for the use of calligraphic interfaces is already in place.

One of the most usual tasks on those devices is text editing. However, common text editors are based on traditional interaction modes, such as sequential text entry, which requires a cursor to be set on the right position. Also, the edition of the text itself is done, for the most part, resorting to menu options or toolbars. Since PDAs and Tablet PCs are much closer, in terms of portability and format, to paper pads, it would benefit users to allow them to manipulate their texts using in the same way they use when working on paper.

The CaliEdit system is an application under development as a year long undergraduate project at IST that combines calligraphic interfaces with a traditional text editor. It allows users to manipulate texts by directly drawing over it symbols representing the most common editing tasks. It recognizes several common symbols users draw on paper to represent desired changes to a text, and effects those changes. CaliEdit uses the CALI gesture recogniser [FONSECA'00], freely available on the Web [CALI] to identify the symbols.

Users played a central role throughout the development of CaliEdit. In an early task analysis phase, several user studies were conducted to discover what are the most common and natural text-editing symbols. Currently, they have been actively engaged in the project through the performance of heuristic and usability evaluations. User-

centred design is of capital importance due to the novel approach to text editing embodied by CaliEdit.

The following section describes how the relevant symbols were identified. Then, the choice of the platform on which CaliEdit was implemented will be described. In the next section several relevant implementation issues will be discussed, followed by some conclusions and references to current and future work.

2. SYMBOLS

Attempts to find a set of standard symbols for text editing both in libraries and on the Internet proved to be unsuccessful. It soon became apparent that there isn't a standard, universal, symbol set for that the correction of paper documents. However, it was desirable to mimic the usual forms of interaction as closely as possible. Determining the most commonly used symbols was of capital importance. Two questionnaire-based studies specially designed to collect the desired information were conducted. Although the first provided interesting results, sometimes a high level of ambiguity (two or three symbols for the same function) didn't allow conclusive results to be extracted. The second questionnaire was created with those ambiguities in mind in order to solve them. The symbols thus obtained are the ones currently in use in CaliEdit.

Before the design of the questionnaires, the set of functions that CaliEdit should perform was defined. Those functions are the most common in text editing:

- delete text
- move text
- insert characters in the middle of words
- insert words in the middle of sentences
- insert paragraphs
- insert tabs
- insert spaces

Evidently, this was an open set that could be changed in the course of the analysis of the questionnaires, if some new relevant tasks came up. As it turned out, it no changes were necessary.

The design of the questionnaires was based on the guidelines available on the web site of the Human Machine Interface course at IST [IHM]. They were composed by two small texts. The first text was flawless and the second was a copy of the first in which several common typing and spelling errors were introduced. The user was asked to correct those errors. In theory, the errors in the text would require the use of all the previously chosen editing functions. Furthermore, all errors were introduced in a way in which correcting one wouldn't affect the correction of others. This was done to avoid confusing the users, and because we were trying to identify symbols associated to only one correcting task.

After analysing both questionnaires, the following symbols were found to be the most common when correcting texts in paper documents:

- To delete text

- To move text

- To insert new character

- To insert new words

- To insert a new paragraph

- To insert a space

/

- To insert a Tab

3. CHOOSING THE PLATFORM

In order to decide in which platform to implement CaliEdit, a comparative study of the three most likely candidates (Palm OS, Pocket PC, and Emacs on a Tablet PC) was performed. This study consisted of an in-depth comparison of all three platforms in order to help choose the one that best fitted the CaliEdit requirements.

The study focused on the text editing capabilities made available by the APIs of the three platforms. The ability to integrate the CALI gesture recognizer into whichever platform was chosen was also studied. The recognizer itself is written in C++ but portability issues could arise, mainly for the PDA platforms, due to their special hardware restrictions. Furthermore, the CALI library receives a set of coordinates, corresponding to the places travelled by the stylus during the drawing of a gesture, returning a list of gesture names, ordered by the probability of corresponding to the given set of the points. The point coordinates must be captured by our application and passed to the recognizer. Hence, the comparison also took into account the capability of each platform to capture high-resolution screen coordinates.

In this study, the Internet was an invaluable source of information, allowing access to relevant documents, manuals, developer environments and code samples. A technical report where more detailed results of this study can be found is located at http://mega.ist.utl.pt/~pmc.

3.1 The PalmOS Platform

The PalmOS is an event driven operating system. As such, all applications have three stages in their execution cycle: the startup, the event loop and the finish. The interaction with the user generates events that are processed in the event loop. Some events are directly related with the pen movements. Using those events it is possible to collect a sample of point coordinates and therefore recognize the gesture/symbol drawn by the user. PalmOS also provides a complete API to manipulate text. Those functions are useful in the implementation of CaliEdit. This platform has some disadvantages mainly because of its small screen size and the need to implement a basic text editor in which to include the calligraphic interface.

Some further experiments were made. A simple application that paints on the screen the points where the stylus touches it was implemented. This small application provided insights on both how to program in the PalmOS system in general, and how to collect and store a set of coordinates to later pass to the recognizer. A version of CALI to the PalmOS system as also developed.

3.2 The Pocket PC Platform

Windows CE is the operating system of the Pocket PC and, like PalmOS, it is an event driven system. Windows CE also provides events that reflect the movements of the pen, and functions that manipulate text. Both platforms are very similar in what can be implemented in them.

The Pocket PC has the same limitations of PalmOS. Additionally, some experimentation showed that it's impossible, apparently, to draw inside a text box. This could pose a serious difficulty in the implementation of CaliEdit, since drawing the symbols provides an important feedback to users and cannot be discarded. Although this problem might have a solution, we didn't spend too many resources to find it.

A Windows CE version of the simple point painting application we created for PalmOS was implemented. This allowed the direct comparisons between the two, to better identify the relevant differences. We concluded that in Windows CE and using the .Net platform to implement applications prototypes can be build much faster than for PalmOS. To integrate the CALI into Windows CE applications, a dynamic link library (dll) was built, allowing applications to easily access the CALI API.

3.3 Emacs on Tablet PC Platform

With Emacs there is no need to implement the text editor. Emacs is extensible thought a special Lisp dialect (Emacs-LISP). To implement CaliEdit, it is fundamental to known the position of the pen. Emacs-LISP, however, only gives access to the coordinates of the text cursor. Those coordinates are measured in characters. The cursor is considered to be located between two characters and not in a specific pixel. So, with Emacs, the major problem is the inability to obtain the information of the pointer device on the pixel level. Several modifications of Emacs' source code to provide the capacity to collect high-resolution coordinates were tried, but all were unsuccessful. Extending Emacs-LISP with a special-purpose function to obtain those coordinates seems impossible, or, at least, not feasible given the resource constraints of this project.

3.4 The Chosen Platform

Overall, the PalmOS platform seemed the best and was chosen to implement CaliEdit. Emacs on Tablet PC was the initial preferred option (mainly due to the versatility provided by a bigger screen) but had to be put aside due to the inability to capture high-resolution point coordinates. Without this capability we can't collect a good sample that represents the symbols drawn by the user. CALI was ported to both the Pocket PC and PalmOS platforms. So, the choice of the PalmOS system was made mainly due to the difficulties in mixing text and drawings on the Pocket PC. Also, more support is available through the PalmOS community.

4. IMPLEMENTATION

Some of the symbols CaliEdit recognizes had to be different from those collected during task analysis. For example, on paper when a character is missing in the middle of a word, people use an arrow with the missing character on top. While on paper this sounds good, in a text editor it's much simpler to use a tap to place the cursor at the desired position and then simply write the character. On paper users can't directly change the text we can do so on a computer. In order to achieve not only a natural interface but also an efficient one, a compromise between what users can do on paper and what they can do on a computer was made.

All text modifying capabilities of CaliEdit were implemented with the PalmOS API as a basis. The CaliEdit itself collects the coordinates of the points that define a gesture, invokes the CALI recognizer to identify that gesture and then chooses the appropriate API calls. The portion of text to be affected by the gesture is computed automatically from special coordinate sets, such as the gesture's bounding box, and the font metrics.

4.1 Adapting CALI to CaliEdit

The algorithm implemented in CALI recognizes simple geometric shapes like triangles or rectangles. The shapes are recognized independently of changes in rotation, size or number of individual strokes. This library was written in C++. The source files of CALI were used instead of the existing pre-compiled library, since it was necessary to port it to other platforms.

There were some problems porting CALI to PalmOS. One of the biggest resulted from the memory limitations inherent to that operating system. The application experienced several stack overflows, especially when recognizing ellipses. This turned out to be the result of the large

number of interpolated ellipse points calculated by the CALI recognizer. These points allow applications that so desire to replace the rough sketch drawn by the user with a more accurate shape. CaliEdit doesn't need this to happen. As a result, the CALI library was altered to stop doing those calculations. Some memory leaks were also found, made visible by PalmOS's memory limitations. Eventually, all known memory leaks were solved, although some may still remain undiscovered.

4.2 Making CaliEdit a Palm-like Application

At first, CaliEdit's user interface was structured as the ones found on text editors on desktop PCs. It soon became obvious that this wasn't the best way to develop a Palm application. Both the the hardware constraints and the ways of interaction are different. A PC-like would be fraught with usability problems and be perceived as strange by a Palm user. Instead, we looked closely at a standard PalmOS text editor, MemoPad, modifying CaliEdit to make it similar to that application.

Overall, the user interface is structured as follows: the first screen presents a list of documents already present on the PDA, from which the user can choose one to modify. Alternatively, a new document can be created with the help of a button. Once a document is selected, the application moves on to the text-editing screen. Again, this is similar to MemoPad. However, two extra buttons inexistent in MemoPad were added to this screen. The first allows users leave the document without saving and the second maintains the original text and saves the modified text in new document, just like the '`Save as…`' function in desktop text editors.

Some standard PalmOS functionalities are present in CaliEdit. It is the case of tapping on a word twice to select the whole word, and thrice to select the line the word is on. The usual PalmOS menus (copy, paste, etc.) were also implemented in CaliEdit. Finally, text can be written using Graffiti, as usual for PalmOS applications.

5. CONCLUSIONS AND FUTURE WORK

A text editor where symbols commonly used on paper to signal required corrections can be used was implemented. One of our goals is to achieve a more natural interaction with text documents. Hence, the next steps in our research will be to perform heuristic and usability evaluations [DIX'97]. Those evaluations will help us find interface problems, and will validate the choices made so far regarding its design.

In the heuristic tests, a Palm with our editor will be made available to 4 persons with knowledge in the HCI field. All errors found by the evaluators will be corrected before the usability tests. In the usability tests we will give users some time to get acquainted with the editor. Then, we will ask them to perform some editing tasks, both using MemoPad then CaliEdit: creating a document, writing something and then saving it, and correcting several errors in a pre-determined text. We intend to interview around 30 persons, half of which will use MemoPad first, while the other half will start with CaliEdit. With this, we hope to prevent biasing the results due to previous knowledge of the texts to correct. Our goal is to show that CaliEdit has better performance than conventional editors like MemoPad. We'll collect information on both the speed and number of errors made by users while performing the tasks. We will also try to discover if users are subjectively more satisfied with CaliEdit than with ordinary editors and with paper and pen. We'll compare the satisfaction levels with the time gained or lost by using CaliEdit. Finally, changes to the interface prompted by the tests will be implemented and validated iteratively.

6. REFERENCES

[CAETANO'02] A. T. Caetano, N. Goulart, M J. Fonseca and J. A. Jorge. JavaSketchIt: Issues in Sketching the Look of User Interfaces. AAAI Spring Symposium on Sketch Understanding, 25-27 March, 2002, Palo Alto, California.

[CALI] The CALI Homepage: http://immi.inesc.pt/cali/

[DIX'97] Alan J. Dix, Janet Finlay, Gregory Abowd and Russel Beale. Human-Computer Interaction, second edition. Prentice Hall, 1997, ISBN 0-13-239864-8

[FONSECA'00] M. J. Fonseca and J. A. Jorge. Using Fuzzy Logic to Recognize Geometric Shapes Interactively. In Proceedings of the 9th Int. Conference on Fuzzy Systems (FUZZ-IEEE 2000). San Antonio, USA, May 2000.

[FONSECA'03] M. J. Fonseca and J. Jorge. Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases. 8th International Conference on Database Systems for Advanced Applications (DASFAA '03), Kyoto, Japan, March 2003.

[IHM] The Human Machine Interface Course at IST site: http://mega.ist.utl.pt/~ic-ihm/

[PEREIRA'03] J. P. Pereira, J. A. Jorge, V.A. Branco, F. N. Ferreira. Calligraphic Interfaces: Mixed Metaphors for Design. Tenth Workshop on the Design, Specification and Verification of Interactive Systems, DSV-IS 2003, Funchal, Portugal, 4-6 June 2003.